EM-basierte maschinelle Lernverfahren für natürliche Sprachen

---

Klassifizierung mehrdimensionaler linguistischer Daten
und
Modellierung lexikalisierter Grammatiken

Von der Fakultät Philosophie der Universität Stuttgart
zur Erlangung der Würde eines Doktors der Philosophie (Dr. phil.)
genehmigte Abhandlung

Vorgelegt von Detlef Prescher aus Sudheim

Hauptberichter: Prof. Ph.D. Mats Rooth
Mitberichter: Prof. Dr. Christian Rohrer
Tag der mündlichen Prüfung: 15. Januar 2002

Institut für Maschinelle Sprachverarbeitung der Universität Stuttgart

Mai 2001

Danksagung

Die vorliegende Abhandlung resultiert zum grössten Teil aus meiner Arbeit als wissenschaftlicher Angestellter am Institut für Maschinelle Sprachverarbeitung (IMS) der Universität Stuttgart, für das ich von Juni 1997 bis Dezember 2000 in verschiedenen Projekten (SPARKLE, VerbMobil, SFB340/B7) tätig war. Letzte wichtige Resultate erzielte ich am Language Technology Lab des Deutschen Forschungszentrums für Künstliche Intelligenz (DFKI GmbH) in Saarbrücken, für das ich seit Januar 2001 als Researcher im MUCHMORE-Projekt arbeite.

Meine Abhandlung wäre nicht ohne die Hilfe, Unterstützung und gemeinsame Arbeit vieler Kolleginnen und Kollegen möglich gewesen.

An erster Stelle möchte ich mich bei Karin Müller bedanken, die mich in meinem Gefühl bestärkte, nach meinem "ersten Leben" als Diplom-Mathematiker (beschäftigt bei der damals grössten deutschen Versicherungsgesellschaft), nun relevante Arbeit auf dem recht jungen Forschungsgebiet der statistischen Computer-Linguistik zu betreiben. Darüber hinaus bedanke ich mich dafür, dass Karin kompetente Partnerin vieler gemeinsamer Forschungsarbeiten war und hoffentlich auch bleiben wird.

Besonders möchte ich mich bei meinem Zweitgutachter, Prof. Christian Rohrer, bedanken, der mir den Rat gab, meine zwei Ausbildungen zum Diplom-Mathematiker und Diplom-Linguisten zu kombinieren, und mich auf das Gebiet der stochastischen Linguistik zu konzentrieren, was damals bedeutete, mich um eine Stelle am Lehrstuhl von Prof. Mats Rooth zu bewerben.

Sehr viel verdanke ich meinem Erstgutachter, Prof. Mats Rooth, der mir den Weg zu den meisten Resultaten dieser Abhandlung wies. Zu grösstem Dank bin ich ihm aber verpflichtet, weil er mir darüberhinaus grosse Freiheiten gab, für mich wichtige Forschungsthemen auch abseits des von ihm gewiesenen Weges zu suchen. Auch hier nahm er Anteil und war, dank seiner unübertrefflichen Intuition, der ich fast bedingunglos vertraue, mein wichtigster Ratgeber. Trotz der neunmonatigen Wartezeit auf meine mündliche Prüfung, die hauptsächlich durch die später eingetretene räumliche Distanz verursacht wurde, bin ich Mats für unsere gemeinsame Zeit sehr dankbar.

Zu Beginn meiner Arbeit an Mats Lehrstuhl hatte ich das Glück, durch Mats und meine damaligen Kollegen Glen Carroll und Marc Light eine erstklassige "amerikanische" Ausbildung in stochastischer Linguistik zu geniessen, die ich heute mit den Attributen "so

klar und geradlinig wie möglich", dabei aber "am bestmöglichen Erfolg orientiert" und mit dem "Verkaufen der geleisteten Arbeit" verbinde.

Zu dieser Gruppe, die wir bald "Gramotron" nannten, kamen später Stefan Riezler, Franz Beil, Helmut Schmid und Sabine Schulte im Walde hinzu. Aus unserer gemeinsamen Arbeit habe ich viel gelernt, besonders aber von Stefan, der mich mit dem EM-Algorithmus bekannt machte. Leider existiert Gramotron nicht mehr, da die meisten Mitglieder inzwischen an anderer Stelle, vor allem in den USA, neue Herausforderungen gesucht haben. Das freundschaftliche Miteinander, mehr noch aber die inspirierende Freude an unserer Arbeit, und auch deren Qualität wird mir als etwas ganz Besonderes in Erinnerung bleiben.

Ausser mit den bisher genannten Kolleginnen und Kollegen habe ich noch mit Uli Heid, Mark Johnson, Jonas Kuhn und Bernd Möbius zusammengearbeitet und publiziert. Bei allen möchte ich mich bedanken, besonders aber bei Bernd, mit dem es eine wirkliche Freude war zusammenzuarbeiten, und der für mich eine ähnliche Rolle wie Mats einnimmt, nur eben auf dem Gebiet der "statistischen Phonologie".

Sehr herzlich möchte ich mich bei Helmut bedanken, der eine frühere Version dieser Abhandlung sorgfältig las, mich auf Fehler hinwies, kritisch kommentierte, aber auch Lob spendete. Helmuts spontane, umfangreiche und kompetente Hilfe kam in einer Zeit, in der ich dies besonders brauchte.

Abschliessend möchte ich mich bei Sabine Schmid (nein, Dieterle!) bedanken, die in all den Jahren und auch jetzt wieder, beim Druck dieser Abhandlung als AIMS-Report, mit Rat und Tat und sonnigem Gemüt weiterhalf.

Inhaltsverzeichnis

Abstract

This thesis presents the *Expectation-Maximization algorithm* (EM algorithm, Dempster et al. (1977)) in its practical and theoretical aspects. The EM algorithm is the stochastic basis of many machine learning algorithms for natural language processing. In the theoretical part of this thesis the stochastic basis of linguistics and the formal basis of the EM algorithm is explained. The practical part of this thesis presents a probabilistic clustering method for multivariate linguistic data and stochastic modeling of lexicalized grammars.

Introduction.

In the first place it is an interesting result of this thesis that most of the stochastic models used by linguists can be interpreted as probabilistic context-free grammars. This result will be accompanied by the formal proof that the Inside-Outside algorithm which is the standard training method for probabilistic context-free grammars, can be regarded as a dynamic-programming variant of the EM algorithm. Even if this result is considered in isolation it is very interesting because it means that most of the probabilistic models used by linguists are trained by a version of the EM algorithm.

This result is even more interesting when considered in a theoretical context because this study has revealed that the well-known convergence behavior of the Inside-Outside algorithm has been confirmed by many experiments but it seems never to have been formally proved. Being a version of the EM algorithm, however, it also possesses its good convergence behavior. The author therefore contends that the as yet imperfect line of argumentation can be transformed into a coherent proof. The focus of this study is, however, the presentation of the formal basis and properties of the EM algorithm and - if necessary - its completion. In this context it is interesting to see that the EM algorithm is a version of a new so-called *stochastic bootstrap algorithm (SB algorithm)* which can be especially well motivated. This algorithm was contrived by the author in order to transform any

stochastic training method working with annotated data (e.g. manually tagged, parsed or semantically annotated text) into an iterative training method which requires only a training corpus of unannotated data (i.e. free text in the most cases) using a symbolic analyser.

A further aim of this study is to present an *EM-based clustering method for multivariate data*. This method is applied to induce semantically annotated lexicons for the identification of idioms, and to disambiguate lexical ambiguities in machine translation and statistical parsing. Finally, this study shows that the EM algorithm can be successfully used for training of accurate stochastic grammars based on free text without using a treebank. Here, two alternative methods are presented: (i) a probabilistic lexicalized context-free grammar, (ii) a stochastic lexicalized unification-based grammar both trained on large text-corpora.

Stochastical Linguistics.

In Chapter 2 the stochastic principles of linguistics are presented. Presently, there are several handbooks providing an introduction into the stochastic principles of natural language processing for linguists. Actually there are a variety of introductions of this kind. Therefore it seems to be reasonable to present in this chapter merely the mathematical terms which are essential for an understanding of the EM algorithm, and its application in the area of linguistics. The terms introduced will be motivated by examples taken from linguistics. Furthermore, mathematical terms are replaced by linguistic vocabulary: Instead of the term "sample", e.g., the term "corpus" will be used. Working on Chapter 2 the author came to the conclusion that all of the popular stochastic models of natural language processing can be interpreted as a specific instance of one single model, i.e. a specific instance of a probabilistic context-free grammar. The main objective of this chapter is therefore the presentation of the best-known stochastic models used in linguistics:

- the language models:
  the n-gram model, the Markov model and Hidden-Markov model, and
- the tree bank grammar.

These models are to be presented in their specific applications. This means more than merely the depiction which training process can be used in order to determine the optimal probability parameter. The objective is rather to show that each of these models can be modelled as a probabilistic context-free grammar.

The EM Algorithm.

Chapter 3 is to amplify the results of Chapter 2 by the formal proof that the training method for probabilistic context-free grammars - the well-known Inside-Outside algorithm (Baker (1979), Lari and Young (1990)) - is in fact a dynamic-programming variant of a specific version of the EM algorithm. If the findings of Chapter 2 are amplified, this finding results in the conclusion that

- the most important stochastic models of natural language processing, the n-gram model, the Markov model, the Hidden-Markov model, the tree bank grammar and the probabilistic context-free grammar can be trained by specific EM algorithms.

Chapter 4 and 5 additionally show that the probabilistic clustering method for multivariate linguistic data, the lexicalized context-free grammars and the stochastic unification-based grammars based on log-linear models are also a version of EM. Therefore it is possible to say (modifying a well-known quotation) that the world is not a finite state machine but an EM algorithm.

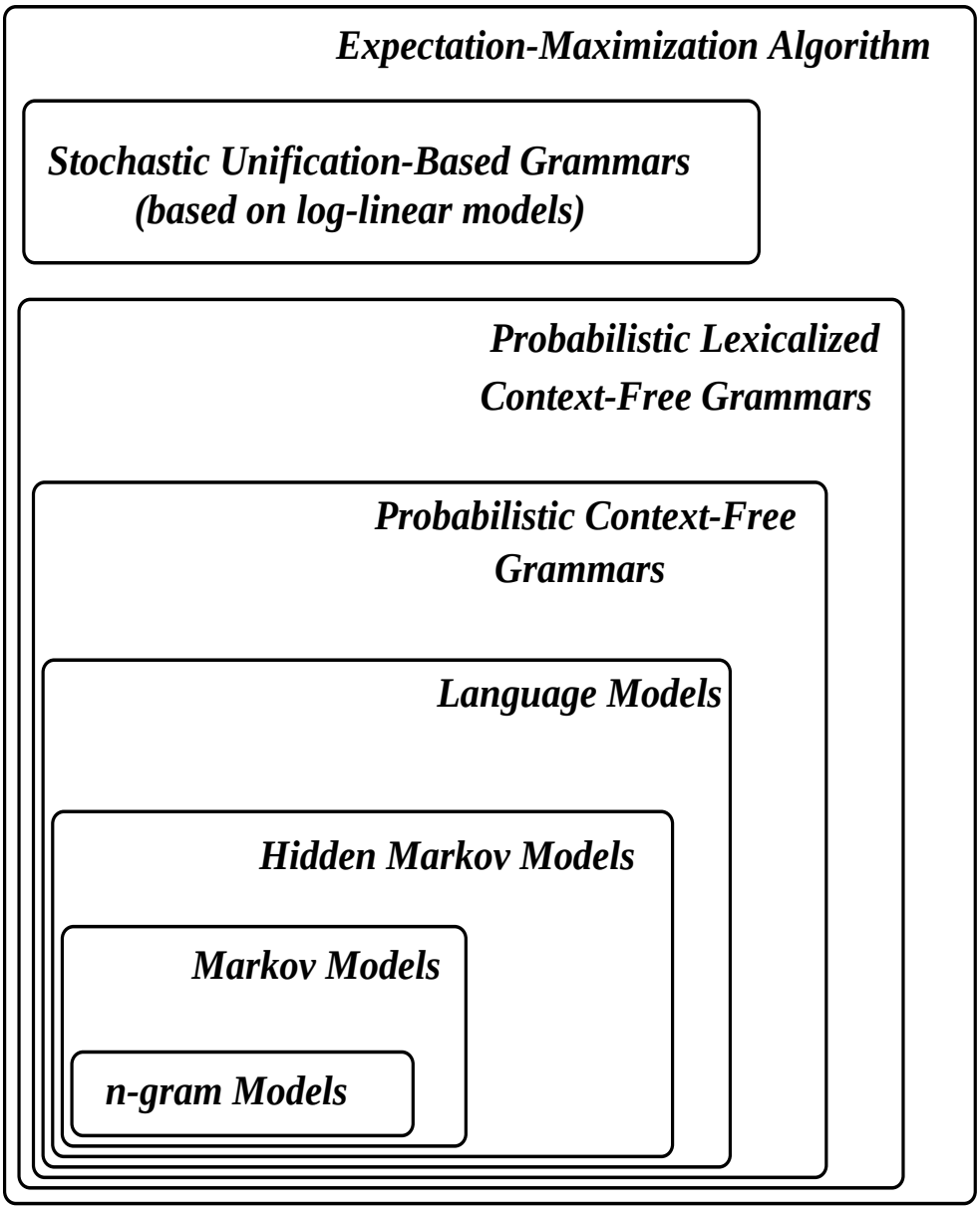


Figure: (Part of the) Linguistic World of EM

A second consequence of the presented work concerns the Inside-Outside algorithm itself. In literature it is often alleged that Baker (1979) or Lari and Young (1990) have proved that the probabilistic grammars when trained by the Inside-Outside algorithm are characterized by monotonously increasing log-likelihood values which converge towards a local maximum of the log-likelihood function. In fact, however, Baker (1979) has only intuitively generalized the Forward-Backward algorithm (Baum 1972), whereas Baum (1972) explicitly stated the Forward-Backward algorithm for Hidden Markov-models. Furthermore, Baker unfortunately studied merely training corpora consisting of one single sentence. Lari and Young (1990) eliminated this weak point and generalized the Inside-Outside algorithm in order to apply it to any training corpus. However, this study also lacks a formal proof of monotonicity and convergence. It is worth noting that Lari and Young (1991) pointed out that certain counts of grammar rules used as in the Inside-Outside algorithm can be interpreted as expected frequencies. Unfortunately, the mathematic principles of EM-theory (Dempster et al. (1977) were not applied to formally proof this intuitively conceived observation.

- To the best of the author's knowledge, the present study formally proves the common wisdom saying the Inside-Outside algorithm is a dynamic-programming variant of the EM algorithm.

Being a dynamic-programming variant of the EM algorithm, the Inside-Outside algorithm also shows the same convergence behavior. This behavior, however, differs in some aspects from its description in literature which is a further finding of this chapter.

- The probabilistic grammars iteratively trained by the Inside-Outside algorithm are characterized by monotonously increasing log-likelihood values (corpus probabilities respectively). A limit point of this sequence is a stationanary point of the log-likelihood function and in specific cases also a local maximum.

The aim of Chapter 3 is to present the formal basis of the EM algorithm and to show how it can be used advantageously in linguistics. This objective also entails the interpretation of the EM algorithm and its study in a larger context. This chapter achieves this objective by interpreting the EM algorithm as a version of a new meta-algorithm which is especially easy to motivate.

- The author developed the so-called *stochastic bootstrap algorithm (SB algorithm)* in order to transform any stochastic training process working with annotated data into an iterative stochastic training process which requires only a training corpus of unannotated data and a symbolic analyzer.

The EM algorithm is a version of the SB algorithm which results from the use of the maximum-likelihood estimation within the SB-algorithm as the basic training process. Unfortunately there is much scepticism about the EM algorithm by many stochastic linguists because in some well-known studies it has been reported that comparatively bad results have been achieved using the EM algorithm. Having a closer look at these studies, however, the reader tends to believe that these bad results were rather caused by experimentalist than by the EM algorithm itself. Therefore, Chapter 3 especially deals with the description of the EM training procedure and the evaluation of the resulting stochastic models in order to enable anyone to obtain better results using EM instead of annotated and extensive corpora.

- The free parameters of the EM algorithm (size of the training corpus of incomplete data, number of training iterations and starting points) are presented and the study suggests how to use a given corpus of incomplete data in order to train optimal parameters and to evaluate them systematically.

In order to apply the EM algorithm to good account, it is not only advantageous to understand how it works but also to know the formal properties of the EM algorithm. At this point it is interesting to note that only two out of four EM theorems given by Dempster et al. (1977) were correct: the theorem of monotonously increasinging lok-likelihood values an the convergence theorem (towards stationary points of the log-likelihood). Wu (1983) added further theorems about convergence towards local maxima and about parameter convergence of the log-likelihood function. The author presents the theory and adds some simple lemmas by his own. Finally, the EM algorithm for probabilistic context-free grammars is presented. The methods and results of the general EM theory are applied to this special case resulting in re-estimation formulas and properties of the EM algorithm for context-free grammars.

EM-Based Clustering of Multivariate Data.

Chapter 4 presents a probabilistic clustering method for multivariate data. The so-called "hard" clustering methods allow the data to belong to no more than one class each, whereas in a probabilistic clustering method any data is allowed to belong to several classes at the same time because class-membership is defined by certain probability parameters. The term "multivariate data" is generally used in a domain with two or more finite sets of objects, vectors are observed in, and each of the vector elements belongs to one of the

object sets. Multivariate data occurs in various applications of natural language processing ranging from phonology to pragmatics. In EM-based clustering for multi-dimensional data, classes are defined as hidden data which are to be learned from a corpus of incomplete data (i.e. data without class annotations). The main task of EM-based clustering for multivariate data therefore is

- the automatic detection of the hidden class structure of data in a given corpus.

Clustering is made feasible by employing a so-called independence assumption. The crucial idea is that within each class any object can be combined with any other forming data types which characterize precisely this class. The independence assumption normally cannot be appropriate for the corpus as a whole, thus the corpus necessarily splits into those parts which will constitute the classes of the model. Generally the number of classes must be experimentally determined to optimally meet the independence assumption. A second task of the EM-based clustering method which is closely connected with the automatic detection of hidden class structure is

- the induction of smoothed probability distributions over the given data.

Unknown data could be defined for a given corpus as those data which do not appear in this corpus. However, they could occure in any virtual corpus which seems reasonable for any virtual application having in mind. By this definition the real data of this world are divided into two parts: (i) the huge number of unknown data (ii) the small number of data occuring in real corpora. Therefore the most important property of all probability models is the ability to cope with unknown data, i.e. the ability to assign a positive probability to unknown (but reasonable) data. EM-based clustering of multivariate data indicates that it can deal sucessfully with unknown data. Firstly, in these models the model probability of a data type is calculated by adding all joint probabilities of this data type and its classes. Thus the data type receives a positive model probability if there is at least one class, with a positive joint probability. Therefore the EM-based clustering models presumably will be smoothed probability distributions. Secondly, the EM-based clustering models show an invariance property. This means that their uniform marginal distributions will not change during the iterative training procedure. Therefore it can be assumed that the EM-based clustering models are not only smoothed probability distributions of empirical probability distributions but also preserve the characteristic properties of the given corpus. Furthermore, Chapter 4 focusses on showing that the EM-based clustering methods can be used in many applications of natural language processing ranging from phonology to pragmatics. Four applications for EM-based clustering methods are presented:

- construction of semantically annotated lexicons,
- identification of verb-noun-collocations,
- lexical disambiguation (in machine translation),
- induction of probabilistic syllable models.

The research presented in this chapter is joint work. Rooth et al. (1998) showed that the probabilistic clustering method for two-dimensional verb-noun pairs suggested by Rooth (ms) is in fact a version of the EM algorithm. The most general formulation of the EM-based clustering method was suggested by Müller et al. (2000a) and it was also successfully used for constructing three- and five-dimensional syllable models. The lexicon induction is studied in Rooth et al. (1999) and Prescher and Heid (2000). The lexical disambiguation method is presented in Prescher et al. (2000), whereas the invariance property is presented in Prescher (2001b). An implementation of EM-based multivariate clustering is available on the Web (Prescher 2000a).

Stochastic Grammars.

Chapter 5 presents stochastic methods for modeling lexicalized grammars. At the moment stochastic grammars which are more powerful than simple probabilistic context-free grammars, are studied intensively. Most of these models incorporate lexical heads in rule based probabilitiy models, e.g. in Collins (1997), Charniak (1997) and Ratnaparkhi (1997). A further aspect of these approaches is that the probability models are trained by treebanks, i.e. corpora of manually disambiguated sentences. The above mentioned approaches use the Penn Wall Street Journal treebank (Baum et al. 1993). This indicates that treebank training is ineffective as it requires manual intervention to assign specific syntax analyses to corpora from specific domains and specific languages. It is generally assumed that only bad results can be achieved using EM training if one does not use at least partial annotations. Experimental results confirming this common wisdom are presented e.g. by Elworthy (1994) and Pereira (1992) for EM training of Hidden Markov models and probabilistic context-free grammars. Chapter 5 is not to discuss these different approaches but to show that the EM algorithm - notwithstanding common wisdom - can be successfully used to train accurate stochastic grammars on free text without using a treebank. Two alternative approaches are presented. Firstly, a probabilistic lexicalized context-free grammar is presented and secondly, a stochastic lexicalzized unification-based

grammar is introduced, both of them trained on large text corpora. The approaches do have strong and weak properties. Probabilistic lexicalized context-free grammars can be trained with the Inside-Outside algorithm, a dynamic-programming variant of the EM algorithm, whereas stochastic lexicalized unification-based grammars can only be trained with non-dynamical EM. Therefore probabilistic lexicalized context-free grammars can be applied and trained effectively. However, the number of analyses has to be controlled with stochastic unification-based grammars. The analyses have to be extremely limited in order to guarantee the practical applicability of the formalism. Unfortunately, this condition is not always met. However, the remarkable good results achieved in experiments with a strictly limited number of analyses (≤ 25) compensates for this disadvantage of stochastic unification-based grammars. It is especially important to note that lexicalization reveals a clearly positive effect to unification-based grammars which is not the case for probabilistic lexicalized context-free grammars. Moreover it was formally proved that for log-linear models (being the basis of stochastic unification-based grammars) there exist "optimal" starting points, which is an important contribution to the EM-theory. This result is important because the general EM algorithm requires the testing of as many starting points as possible. To the best of the author's knowledge this is the first publication of a theoretical explanation for the choice of concrete fixed starting points. The experiments described in Chapter 5 is joint work. The experiments concerning probabilistic context-free grammars are published in Beil et al. (1999), whereas the experiments concerning stochastic unification-based grammars are described in Riezler et al. (2000).

Summary.

Summarizing the above, it could be said that the present study provides in its first part an introduction to stochastic linguistics. It presents the mathematic basis of the EM algorithm and describes the EM algorithm within the context of the stochastic bootstrap-algorithm and it shows that the best-known machine learning algorithms in linguistics are simply specific instances of dynamic-programming variants of the EM algorithm. In the second part, an EM-based clustering method for multivariate data is presented resulting in various linguistic applications ranging from phonology to pragmatics. Furthermore probabilistic context-free grammars and stochastic unification-based grammars are presented, their strong and weak points, the stochastic components of which can be trained due to the EM algorithm on large free text copora yielding accurate stochastic parser.

Kapitel 1

Einführung

In dieser Arbeit wird der Erwartungswert-Maximierungs-Algorithmus (EM-Algorithmus, Dempster et al. (1977)), der die stochastische Grundlage vieler maschineller Lernverfahren für natürliche Sprachen ist, in Theorie und Praxis vorgestellt. Im theorieorientierten Teil dieser Arbeit werden die stochastischen Grundlagen der Linguistik und die formalen Grundlagen des EM-Algorithmus vorgestellt. Der praxisorientierte Teil dieser Arbeit beschäftigt sich mit einem probabilistischen Klassifikationsverfahren für mehrdimensionale linguistische Daten, sowie mit der Modellierung von stochastischen lexikalisierten Grammatiken.

Ein erstes interessantes Ergebnis dieser Arbeit ist, dass die meisten der in der Linguistik benutzten stochastischen Modelle als probabilistische kontextfreie Grammatiken interpretiert werden können. Dieses Ergebnis wird in dieser Arbeit dahingehend ergänzt werden, dass formal bewiesen wird, dass der Inside-Outside-Algorithmus, der das Standard-Trainingsverfahren für probabilistische kontextfreie Grammatiken ist, als eine dynamische Programmiervariante des EM-Algorithmus aufgefasst werden kann. Dieses Ergebnis ist für sich allein genommen schon interessant, da es besagt, dass die meisten der in der Linguistik benutzten stochastischen Modelle mit einer Instanz des EM-Algorithmus trainiert werden. Darüber hinaus ist das Ergebnis von grösserem theoretischem Interesse, da im Verlaufe dieser Arbeit entdeckt wurde, dass das allseits bekannte monotone Konvergenzverhalten des Inside-Outside-Algorithmus zwar in vielen Experimenten bestätigt ist, aber scheinbar noch niemals formal bewiesen wurde. Als Instanz des EM-Algorithmus erbt der Inside-Outside-Algorithmus aber dessen gutes Konvergenzverhalten, sodass auf diesem Weg die (nach bestem Wissen des Verfassers) vorhandene Beweislücke geschlossen werden kann. Der Schwerpunkt dieser Arbeit besteht allerdings darin, die formalen Grundlagen und Eigenschaften des EM-Algorithmus vorzustellen und, wenn nötig, zu ergänzen. Hierbei ist interessant, dass der EM-Algorithmus eine besonders gut motivierbare Instanz

eines neuen *stochastischen Bootstrap-Algorithmus (SB-Algorithmus)* ist. Dieser wurde in der vorliegenden Dissertation konzipiert, um ein beliebiges stochastisches Trainingsverfahren, das mit *annotierten Datentypen* arbeitet (z.B. mit manuell getaggten, geparsten oder semantisch annotierten Text), mit Hilfe einer *symbolischen Analyse-Komponente* in ein iteratives stochastisches Trainingsverfahren zu überführen, welches lediglich einen *Trainingskorpus von unannotierten Datentypen* benötigt (meistens freien Text). Ein weiterer Schwerpunkt dieser Arbeit ist die Vorstellung eines EM-basierten Klassifikationsverfahrens für *mehrdimensionale linguistische Datentypen*, und dessen Anwendung zum Aufbau eines semantisch annotierten Lexikons, zur Identifikation von Kollokationen, sowie zur Disambiguierung lexikalischer Ambiguitäten in der maschinellen Übersetzung und statistischem Parsing. Abschliessend wird in dieser Arbeit gezeigt werden, dass der EM-Algorithmus, im Gegensatz zur weit verbreiteten Meinung, mit Erfolg eingesetzt werden kann, um akurate stochastische Grammatiken auf freiem Text, ohne Benutzung einer Baumbank, zu trainieren. Hierbei werden zwei alternative Ansätze vorgestellt: zum einen wird eine probabilistische lexikalisierte kontextfreie Grammatik vorgestellt, zum anderen eine stochastische[1] lexikalisierte unifikationsbasierte Grammatik, die aber beide auf grossen Textkorpora trainiert werden.

In Kapitel 2 werden die stochastischen Grundlagen der Linguistik vorgestellt. Für Linguisten stehen inzwischen einige Einführungen in die stochastischen Grundlagen der maschinellen Sprachverarbeitung zur Verfügung. Tatsächlich ist das Angebot sogar so reichhaltig, dass es sinnvoll erschien, in diesem Kapitel nur die mathematischen Begriffe vorzustellen, die für das Verständnis des EM-Algorithmus und seiner Anwendungen innerhalb der Linguistik unbedingt notwendig sind. Die eingeführten Begriffe werden mit Beispielen aus der Linguistik motiviert und es werden Bezeichnungen verwendet, die in der Linguistik üblich sind: zum Beispiel wird anstelle des Begriffs *Stichprobe* der Begriff *Korpus* verwendet. Im Laufe der Fertigstellung des 2. Kapitels kristallisierte sich heraus, dass sämtliche populären stochastischen Modelle der maschinellen Sprachverarbeitung als Spezialfall eines einzigen Modells angesehen werden können, nämlich als Spezialfall einer *probabilistischen kontextfreien Grammatik*. Das Hauptziel von Kapitel 2 ist daher, die bekanntesten stochastischen Modelle der Linguistik,

- die *Sprachmodelle*:
  das *n-gram Modell*, das *Markov-Modell*, und das *Hidden-Markov-Modell*,
- die *Baumbank-Grammatik*,

[1] Der Term "probabilistisch" wird für unifikationsbasierte Grammatiken vermieden, da diese mit einem stochastischen Modell versehen werden, das keine Wahrscheinlichkeitsverteilung auf der Menge aller grammatischen Analysen liefert.

möglichst in einer spezifischen Anwendung vorzustellen und nicht nur zu zeigen, mit welchen *Trainingsverfahren* optimale *Wahrscheinlichkeitsparameter* bestimmt werden können, sondern zusätzlich vorzustellen, dass jedes dieser Modelle als eine probabilistische kontextfreie Grammatik modelliert werden kann.

In Kapitel 3 soll die Arbeit aus Kapitel 2 ergänzt werden, indem formal bewiesen wird, dass das Trainingsverfahren der probabilistischen kontextfreien Grammatiken, der bekannte *Inside-Outside-Algorithmus* (Baker (1979), Lari und Young (1990)), eine *dynamische Programmiervariante* einer speziellen Instanz des *EM-Algorithmus* ist. Unter Benutzung der Ergebnisse aus Kapitel 2 führt dieses Ergebnis zu einer ersten Folgerung:

- Die wichtigsten stochastischen Modelle der maschinellen Sprachverarbeitung, das *n-gram Modell*, das *Markov-Modell*, das *Hidden-Markov-Modell*, die *Baumbank-Grammatik* und die *probabilistische kontextfreie Grammatik* können mit speziellen EM-Algorithmen trainiert werden.

Später (Kapitel 4 und 5) wird gezeigt werden, dass sich zu diesen Modellen noch ein *Klassifizierungsverfahren für mehrdimensionale linguistische Datentypen* und die *lexikalisierten kontextfreien Grammatiken*, sowie die auf *log-linearen Modellen* basierenden *stochastischen unifikationsbasierten Grammatiken* gesellen werden. "Die Welt" ist daher, in Abwandlung eines Zitats, kein *endlicher Automat*, sondern ein EM-Algorithmus.

Eine zweite Konsequenz betrifft den Inside-Outside-Algorithmus selbst. In der Literatur wird oft behauptet, dass Baker (1979) oder Lari und Young (1990) bewiesen hätten, dass die mit dem Inside-Outside-Algorithmus iterativ trainierten probabilistischen Grammatiken *mononton wachsende Log-Likelihoodwerte* (bzw. Korpus-Wahrscheinlichkeiten) aufweisen, und gegen ein *lokales Maximum der Log-Likelihood-Funktion* konvergieren. Richtig ist jedoch, dass Baker (1979) lediglich eine intuitive Verallgemeinerung des *Forward-Backward-Algorithmus* (Baum 1972) vorgenommen hat und Baum (1972) den formalen "Monotonie- und Konvergenzbeweis" explizit für Hidden-Markov-Modelle angab. Leider kommt hinzu, dass Baker (1979) nur Trainingskorpora betrachtete, die aus einem einzigen Satz bestehen. Lari und Young (1990) beseitigten diese Schwäche und verallgemeinerten den Inside-Outside-Algorithmus, um ihn für beliebige Trainingskorpora anwenden zu können. Auch in dieser Arbeit fehlt aber ein formaler Monotonie- und Konvergenzbeweis. Es ist interessant, dass Lari und Young (1991) darauf hinweisen, dass gewisse im Inside-Outside-Algorithmus benutzte *Grammatikregel-Counts* als *erwartete Grammatikregelhäufigkeiten* interpretiert werden können. Leider wendeten sie aber nicht die mathematischen Grundlagen der EM-Theorie an, um diese *intuitiv erfasste Schlüsselbeobachtung* in ihrer Arbeit auch *formal zu beweisen.*

- Nach bestem Wissen des Verfassers wird daher die allgemein bekannte Ansicht, dass der Inside-Outside-Algorithmus eine dynamische Programmiervariante des EM-Algorithmus ist, zum ersten Mal in der vorliegenden Arbeit formal bewiesen.

Als *dynamische Programmiervariante des EM-Algorithmus* erbt der Inside-Outside-Algorithmus alle seine Konvergenzeigenschaften. Diese stimmen allerdings nicht völlig mit den Eigenschaften überein, die man gemeinhin dem Inside-Outside-Algorithmus zuschreibt, und stellen somit ein weiteres wichtiges Ergebnis von Kapitel 3 dar:

- Die mit dem Inside-Outside-Algorithmus iterativ trainierten probabilistischen Grammatiken weisen *mononton wachsende Log-Likelihoodwerte* (bzw. Korpus-Wahrscheinlichkeiten) auf, und ein Grenzwert dieser Folge ist ein *stationärer Wert der Log-Likelihood-Funktion* und in gewissen Fällen auch ein *lokales Maximum*.

Das Hauptziel von Kapitel 3 besteht darin, die formalen Grundlagen des EM-Algorithmus vorzustellen und zu zeigen, wie er in der Linguistik nutzbringend angewendet werden kann. Dazu gehört, eine Interpretation des EM-Algorithmus anzugeben und den EM-Algorithmus in einem grösseren Kontext zu stellen. Dies gelingt, indem der EM-Algorithmus als eine besonders gut motivierbare Instanz eines neuen *Meta-Algorithmus* vorgestellt wird.

- Der *stochastische Bootstrap-Algorithmus (SB-Algorithmus)* wurde in dieser Dissertation konzipiert, um ein beliebiges stochastisches Trainingsverfahren, das mit *annotierten Datentypen* arbeitet, mit Hilfe einer *symbolischen Analyse-Komponente* in ein iteratives stochastisches Trainingsverfahren zu überführen, welches lediglich einen *Trainingskorpus von unannotierten Datentypen* benötigt.

Der EM-Algorithmus ist eine Instanz des SB-Algorithmus, die man erhält, indem im SB-Algorithmus als zugrundeliegendes Trainingsverfahren die *Maximum-Likelihood-Estimierung* benutzt wird. Leider wird dem EM-Algorithmus in der stochastischen Linguistik immer noch mit Skepsis begegnet, da in einigen bekannteren Arbeiten relativ schlechte Ergebnisse reportiert wurden, die mit dem EM-Algorithmus erzielt wurden. Nach einem schärferen Blick auf viele dieser Arbeiten kann aber vermutet werden, dass die schlechten Ergebnisse eher dem Experimentator als dem EM-Algorithmus zuzuschreiben sind. Ein besonderes Anliegen von Kapitel 3 ist daher, das Training mit dem EM-Algorithmus und die Evaluierung der erhaltenen stochastischen Modelle so zu beschreiben, dass jedermann in die Lage versetzt wird, mit dem EM-Algorithmus möglichst bessere Ergebnisse zu erzielen, als mit alternativen und teuren Trainingsverfahren, die auf manuell erstellten Trainingskorpora basieren.

- Die freien Parameter des EM-Algorithmus (*Grösse des Trainingskorpus unvollständiger Datentypen*, *Anzahl Trainingsiterationen* und *Startparameter*) werden vorgestellt und es wird vorgeschlagen, wie ein gegebener *Korpus von unvollständigen Datentypen* dazu benutzt werden kann, diese Parameter optimal einzustellen und systematisch zu evaluieren.

Für eine nutzbringende Anwendung des EM-Algorithmus ist es nicht nur vorteilhaft, zu verstehen, wie der EM-Algorithmus arbeitet, sondern auch zu wissen, welche formalen Eigenschaften der EM-Algorithmus besitzt. Hierbei ist interessant, dass von den vier Theoremen, die Dempster et al. (1977) bei der Einführung des EM-Algorithmus angaben, nur zwei Theoreme korrekt waren, nämlich die Theoreme zur *wachsenden Monotonie* und zur *Konvergenz (gegen stationäre Werte)* der Log-Likelihood. Wu (1983) ergänzte diese Aussagen um Aussagen zur *Konvergenz gegen lokale Maxima*, sowie zur *Parameter-Konvergenz der Log-Likelihood*. In dieser Dissertation werden diese Ergebnisse vorgestellt und um einige eigene einfache Aussagen ergänzt. Daran anschliessend werden die vorgestellten Techniken und Ergebnisse des 3. Kapitels auf die *probabilistischen kontextfreien Grammatiken* angewendet, um den EM-Algorithmus für kontextfreie Grammatiken und seine Eigenschaften vorzustellen.

In Kapitel 4 wird ein *probabilistisches Klassifikationsverfahren* für *mehrdimensionale linguistische Datentypen* vorgestellt. Im Gegensatz zu sogenannten "harten Klassifikationsverfahren", die jeden Datentyp in genau eine Klasse stecken, kann in einem probabilistischen Klassifikationsverfahren jeder Datentyp prinzipiell zu mehreren Klassen gehören, da die Klassenzugehörigkeit über gewisse Wahrscheinlichkeitsparameter geregelt wird. Der Begriff *mehrdimensionaler Datentyp* wird im allgemeinen in einer *Domäne* mit zwei oder mehreren endlichen Objektmengen benutzt, in welcher *Vektoren* beobachtet werden, und jedes der Vektorelemente aus einer der Objektmengen stammt. Mehrdimensionale Datentypen kommen in vielen Anwendungen der maschinellen Sprachverarbeitung und in allen Bereichen, von der Phonologie bis zur Pragmatik, vor. Im sogenannten *EM-basierten Klassifikationsverfahren für mehrdimensionale Datentypen* werden *Klassen* als *versteckte Datentypen* angesehen, die aus einem Korpus von *unvollständigen Datentypen* (d.h. von Daten ohne Klassen-Annotationen) unter Benutzung des EM-Algorithmus gelernt werden sollen. Die Hauptaufgabe des EM-basierten Klassifikationsverfahrens für mehrdimensionale Datentypen ist daher

- die *automatische Enthüllung der verborgenen Klassenstruktur* der Datentypen in einem gegebenen Korpus.

Die Klassifizierung der Datentypen wird durch eine sogenannte *Unabhängigkeitsannahme*

möglich gemacht. Als Schlüsselidee wird hierbei angenommen, dass innerhalb einer jeden Klasse alle Objekte miteinander frei kombinierbar sind, um Datentypen zu bilden, die genau für diese Klasse charakteristisch sind. Da die Unabhängigkeitsannahme in der Regel nicht für den gesamten Korpus angemessen ist, hat sie zur Folge, dass der Korpus in Teile zerbrechen muss, welche die Klassen des Modells bilden werden. Im allgemeinen muss die Anzahl der Klassen experimentell bestimmt werden, um zu erreichen, dass die Unabhängigkeitsannahme optimal getroffen wird. Eine zweite, mit dieser Aufgabe eng verbundene Aufgabe des EM-basierten Klassifikationsverfahrens ist

- die *Induktion einer glatten Wahrscheinlichkeitsverteilung* über den gegebenen Datentypen.

Für einen gegebenen Korpus, können *unbekannte Daten* als diejenigen Datentypen definiert werden, die nicht in diesem Korpus vorkommen, aber in irgendeinem virtuellen Korpus, der für irgendeine virtuelle Anwendung sinnvoll erscheint. Mit dieser Definition werden die realen Daten dieser Welt in zwei Teile zerlegt: (i) die riesige Menge unbekannter Daten und (ii) die winzige Menge der in realen Korpora vorliegenden Daten. Aus diesem Grund ist die wichtigste Eigenschaft aller Wahrscheinlichkeitsmodelle ihre Fähigkeit mit unbekannten Daten umzugehen, d.h. unbekannten (aber sinnvollen) Datentypen eine positive Wahrscheinlichkeit zuzuweisen. Es weist vieles darauf hin, dass EM-basierte Klassifikationsmodelle diese Eigenschaft in besonders vorteilhafter Weise realisieren. Zum einen berechnet sich innerhalb dieser Modelle die Wahrscheinlichkeit eines Datentyps als Summe der Wahrscheinlichkeiten dieses Datentyps gegeben seiner Klassen. Ein Datentyp muss also lediglich innerhalb einer einzigen Klasse eine positive Wahrscheinlichkeit besitzen, um eine positive Modell-Wahrscheinlichkeit aufzuweisen, sodass die EM-basierten Klassifikationsmodelle voraussichtlich glatte Wahrscheinlichkeitsverteilungen sind. Zum anderen weisen die EM-basierten Klassifikationsmodelle eine *Invarianzeigenschaft* auf, die besagt, dass sich ihre *Marginalverteilungen*[2] während der iterativen Bestimmung der Wahrscheinlichkeitsparameter nicht ändern wird. Es kann also angenommen werden, dass die EM-basierten Klassifikationsmodelle nicht nur glatte Wahrscheinlichkeitsverteilungen der empirischen Wahrscheinlichkeitsverteilung sind, sondern zusätzlich die charakteristischen Eigenschaften des gegebenen Korpus behalten werden. Ein weiterer Schwerpunkt des 4. Kapitels ist, zu zeigen, dass die EM-basierten Klassifikationsmodelle in vielen Anwendungen der maschinellen Sprachverarbeitung, von der Phonologie bis zur Pragmatik, mit Erfolg eingesetzt werden können. Es werden vier Anwendungen für EM-basierte Klassifikationsmodelle vorgestellt:

[2]Die Wahrscheinlichkeitsverteilungen der einzelnen Koordinaten der gegebenen mehrdimensionalen Datentypen werden *Marginal- oder Randverteilungen* genannt.

- Aufbau semantisch annotierter Lexika,
- Identifikation von Verb-Nomen-Kollokationen,
- Lexikalische Disambiguierung (in der maschinellen Übersetzung),
- Induktion probabilistischer Silbenmodelle

Die in Kapitel 4 vorgestellten Anwendungen sind zum grössten Teil das Ergebnis gemeinsamer Arbeiten, unter wesentlicher Mitwirkung des Verfassers. In Rooth et al. (1998) wird gezeigt, dass das in Rooth (1995) vorgestellte probabilistische Klassifikationsverfahren für zwei-dimensionale Verb-Nomen-Paare tatsächlich eine Instanz des EM-Algorithmus ist[3]. In seiner allgemeinsten Form wurde das EM-basierte Klassifikationsverfahren in Müller et al. (2000a) vorgestellt und erfolgreich zum Aufbau drei- und fünfdimensionaler Silbenmodelle benutzt. Der Lexikonaufbau wird in Rooth et al. (1999) und Prescher und Heid (2000) und die Disambiguierungsmethode in Prescher et al. (2000) vorgestellt. Einige neue theoretische und praktische Eigenschaften der EM-basierten Klassifikationsmodelle werden in Prescher (2001d) vorgestellt. Für Forschungszwecke steht eine Implementierung des EM-basierten Klassifikationsverfahrens (für mehrdimensionale Datentypen) auf dem Web zur freien Verügung (Prescher 2001a).

Kapitel 5 beschäftigt sich mit der Modellierung von stochastischen lexikalisierten Grammatiken. Gegenwärtig werden stochastische Grammatiken, die von prinzipiell grösserer Mächtigkeit als die probabilistischen kontextfreien Grammatiken sind, intensiv erforscht. Eine gemeinsame Eigenschaft der meisten dieser Modelle ist die Berücksichtigung von lexikalischen Köpfen in regelbasierten Wahrscheinlichkeitsmodellen, etwa in Collins (1997), Charniak (1997) und Ratnaparkhi (1997). Ein weiterer Aspekt dieser Ansätze ist, dass die Wahrscheinlichkeitsmodelle mit Baumbanken trainiert werden, d.h. mit Korpora von manuell annotierten Sätzen. In allen zitierten Ansätzen wird die Penn-Wall-Street-Journal-Baumbank (Marcus et al. 1993) benutzt, was ein Zeichen dafür ist, dass das Baumbank-Training aufwendig ist, da es die erhebliche manuelle Intervention erfordert, Korpora aus speziellen Domänen und Sprachen mit speziellen Syntaxanalysen zu versehen. Es wird allgemein angenommen, dass mit dem Training mit dem EM-Algorithmus nur schlechte Resultate zu erzielen sind, wenn nicht zumindest partielle Annotationen verwendet werden. Experimentelle Resultate, die dieses "Allgemeinwissen" bestärken, werden z.B. in Elworthy (1994) und Pereira und Schabes (1992) für das EM-Training von Hidden-Markov-Modellen und probabilistischen kontextfreien Grammatiken präsentiert.

[3] Pereira et al. (1993) und Hofmann und Puzicha (1998) benutzen dasselbe zweidimensionale Modell, aber andere statistische Inferenzmethoden.

In Kapitel 5 sollen nicht diese verschiedenen Ansätze diskutiert werden, sondern es soll vielmehr gezeigt werden, dass der EM-Algorithmus, im Gegensatz zur weit verbreiteten Meinung, mit Erfolg eingesetzt werden kann, um akurate stochastische Grammatiken auf freiem Text, ohne Benutzung einer Baumbank, zu trainieren. Hierbei werden zwei alternative Ansätze vorgestellt: zum einen wird eine probabilistische lexikalisierte kontextfreie Grammatik vorgestellt, zum anderen eine stochastische lexikalisierte unifikationsbasierte Grammatik, die aber beide auf grossen Textkorpora trainiert werden. Beide Experimente sind zum grössten Teil das Ergebnis gemeinsamer Arbeit. Das Experiment zu den probabilistischen kontextfreien Grammatiken wurde in Beil et al. (1999) veröffentlicht, während die Experimente mit der stochastischen unifikationsbasierten Grammatik in Riezler et al. (2000) beschrieben werden. Beide Ansätze haben spezifische Stärken und Schwächen. Während probabilistische lexikalisierte kontextfreie Grammatiken mit dem Inside-Outside-Algorithmus, einer dynamischen Programmiervariante des EM-Algorithmus trainiert werden können, muss für das Training von stochastischen lexikalisierten unifikationsbasierten Grammatiken auf eine nicht-dynamische Instanz des EM-Algorithmus zurückgegriffen werden. Dies hat zur Folge, dass die probabilistischen lexikalisierten kontextfreien Grammatiken effektiv trainiert und angewendet werden können, während es bei den stochastischen lexikalisierten unifikationsbasierten Grammatiken unerlässlich ist, die Anzahl der Analysen eines Satzes stark zu kontrollieren, d.h. extrem zu beschränken, damit der Formalismus anwendbar ist. Leider wird diese Eigenschaft nicht immer erfüllt sein. Der Nachteil der stochastischen lexikalisierten unifikationsbasierten Grammatiken wird aber möglicherweise dadurch ausgeglichen, dass sie in Experimenten, in denen die Anzahl der durchschnittlichen Analysen stark beschränkt war (≤ 25), sehr gute linguistische Ergebnisse zeigten. Insbesondere konnte gezeigt werden, dass die vorgenommene Lexikalisierung einen deutlich positiven Effekt hatte, was bei den probabilistischen lexikalisierten kontextfreien Grammatiken nicht der Fall war. Ferner konnte als wichtiger theoretischer Beitrag zur EM-Theorie formal begründet werden, dass für log-lineare Modelle (als Grundlage der stochastischen unifikationsbasierten Grammatiken) "optimale" *Startparameter* existieren. Dieses Ergebnis ist wichtig, weil der allgemeine EM-Algorithmus im Gegensatz hierzu anweist, möglichst viele Startparameter auszuprobieren. Nach bestem Wissen des Verfassers ist dies das erste Mal, dass eine theoretische Begründung für die Wahl bestimmter, fester Startparameter angegeben wird.

Zusammenfassend bietet die vorliegende Arbeit im ersten Teil eine Einführung in die stochstische Linguistik, stellt die mathematischen Grundlagen des EM-Algorithmus vor, stellt den EM-Algorithmus in den Kontext des neuen *stochastischen Bootstrap-Algorithmus* und zeigt, dass die bekanntesten maschinellen Lernverfahrem der Linguistik einfache Spezialfälle oder dynamische Programmiervarianten des EM-Algorithmus sind.

Im zweiten Teil wird ein EM-basiertes Klassifikationsverfahren für mehrdimensionale Datentypen vorgestellt, das zu einer Reihe von linguistischen Anwendungen führt, die von der Phonologie bis zur Pragmatik reichen. Ferner werden probabilistische lexikalisierte kontextfreie Grammatiken und stochastische unifikationsbasierte Grammatiken mit ihren spezifischen Stärken und Schwächen vorgestellt, deren stochastischen Komponenten dank des EM-Algorithmus auf grossen Trainingskorpora, ohne Benutzung annotierter Korpora, trainiert werden können und akurate stochastische Parser liefern.

Kapitel 2

Stochastische Linguistik

Für das Studium des EM-Algorithmus sind mathematische Kenntnisse aus der Analysis, der Algebra, der Wahrscheinlichkeitstheorie, der Statistik und der Informationstheorie nötig. Auch für Linguisten stehen inzwischen einige Einführungen in die stochastischen Grundlagen der maschinellen Sprachverarbeitung zur Verfügung. Tatsächlich ist das Angebot sogar so reichhaltig, dass es sinnvoll erschien, in diesem Kapitel nur die mathematischen Begriffe vorzustellen, die für das Verständnis des EM-Algorithmus und seiner Anwendungen innerhalb der Linguistik unbedingt notwendig sind. Die eingeführten Begriffe werden mit Beispielen aus der Linguistik motiviert und es werden Bezeichnungen verwendet, die in der Linguistik üblich sind: zum Beispiel wird anstelle des Begriffs *Stichprobe* der Begriff *Korpus* verwendet.

Im Laufe der Fertigstellung dieses Kapitels kristallisierte sich heraus, dass sämtliche populären stochastischen Modelle der maschinellen Sprachverarbeitung als Spezialfall eines einzigen Modells angesehen werden können, nämlich als Spezialfall einer *probabilistischen kontextfreien Grammatik*. Das wichtigere Ziel dieses Kapitels ist daher, die bekanntesten stochastischen Modelle der Linguistik,

- das *n-gram Modell*,
- das *Markov-Modell*,
- das *Hidden-Markov-Modell*,
- die *Baumbank-Grammatik*,

möglichst in einer spezifischen Anwendungen vorzustellen und zu zeigen, mit welchen *Trainingsverfahren* optimale *Wahrscheinlichkeitsparameter* bestimmt werden können. Andererseits wird explizit gezeigt, dass jedes dieser Modelle als eine probabilistische kontextfreie

Grammatik modelliert werden kann. Dieses Kapitel gliedert sich in die folgenden Abschnitte:

Abschnitt 2.1 führt die Begriffe *Wahrscheinlichkeit*, *Wahrscheinlichkeitsverteilung*, *Korpus*, *Datentoken*, *Datentyp*, sowie die zentralen Begriffe der *Korpuswahrscheinlichkeit*, der *Häufigkeitsverteilung* und der *empirischen Wahrscheinlichkeitsverteilung* ein. Es wird gezeigt, dass ein Korpus mit der Häufigkeitsverteilung (oder der empirischen Wahrscheinlichkeitsverteilung) der Datentypen im Korpus identifizierbar ist.

Abschnitt 2.2 führt den Begriff des *linguistischen Ereignisses* und der *Wahrscheinlichkeitsverteilung auf der Menge der linguistischen Ereignisse* ein. Damit werden die Begriffe *probabilistische kontextfreie Grammatik*, *Grammatikregel-Wahrscheinlichkeit*, *Syntaxbaum-Wahrscheinlichkeit* und *Satz-Wahrscheinlichkeit* eingeführt. Es wird motiviert, dass die Grammatikregel-Wahrscheinlichkeiten eine *Standard-Nebenbedingung* erfüllen sollten, damit Grammatiken *konsistent* sind. Unter Benutzung eines fast vergessenen Theorems wird bewiesen, dass *probabilistisches mehrdimensionales Clustering* mit einer probabilistischen kontextfreien Grammatik konsistent ist.

In Abschnitt 2.3 wird der Begriff des *Erwartungswerts* und seine zwei(!) *Linearitäts-Eigenschaften* vorgestellt. Es wird gezeigt, dass sich die Korpuswahrscheinlichkeit in die Erwartungswerte *Log-Likelihood*, *Cross-Entropie*, und *Kullback-Leibler-Distanz*, sowie in die *Korpusperplexität* umrechnen lässt. Es wird motiviert, dass die Korpusperplexität von allen diesen Metriken die beste ist. Die fundamentale *Informationsungleichung* der Informationstheorie wird vorgestellt: sie besagt, dass die *relative Entropie* nicht-negativ ist, bzw. dass die Cross-Entropie mindestens so gross ist wie die *Entropie*.

In Abschnitt 2.4 wird die *Maximum-Likelihood-Estimierung*, die *Maximum-A-Posteriori-Estimierung* und ein Streit zwischen Stochastikern vorgestellt. Es werden *Parametrisierungen*, *Parameter*, und *Parameterräume* eingeführt. Ausserdem wird ein neuer(!) Beweis vorgeführt, der zeigt, dass die empirische Wahrscheinlichkeitsverteilung die Korpuswahrscheinlichkeit maximiert. Es wird ausgeführt, dass die hierfür in der Literatur oft benutzte *Multiplikatorregel von Lagrange* nicht ausreicht. Natürlich wird auch darauf hingewiesen, dass dies ein neuer Beweis der Informationsungleichung der Informationstheorie ist.

In Abschnitt 2.5 wird das *Baumbank-Trainingsverfahren* und seine *Evaluierung* diskutiert. Es werden *Baumbanken* und *Baumbank-Grammatiken* an einem Beispiel vorgestellt. Es wird ein neuer(!) Beweis vorgeführt, der zeigt, dass die *Baumbank-Wahrscheinlichkeit* von der Baumbank-Grammatik unter Standard-Nebenbedingung maximiert wird. Der Schlüsselschritt dieses neuen Beweises ist, dass die *Baumbank-Wahrscheinlichkeit* ein Wahrscheinlichkeit-Produkt gewisser *Teilkorpora* ist, von denen bekannt ist, dass sie durch ihre jeweiligen empirischen Wahrscheinlichkeitsverteilungen maximiert werden.

In Abschnitt 2.6 wird die Unterscheidung in *symbolische* und *stochastische Parsingverfahren* vorgenommen. Es werden sehr ausführlich zwei symbolische Parsingverfahren: *Parsewald-*, und *CKY-Algorithmus*, sowie drei stochastische Parsingverfahren: *Count-*, *Inside-*, und *Viterbi-Algorithmus* vorgestellt. Für alle Algorithmen wird sorgfältig herausgestellt, dass sie *dynamische Programmierung* verwenden. Es wird gezeigt, dass alle Algorithmen *effizient* sind, nämlich von der *Ordnung* $O(n^3)$, wobei n die Satzlänge ist.

In Abschnitt 2.7 wird der *Semiring-Parsing-Algorithmus* (Goodman 1998) vorgestellt, der alle in Abschnitt 2.6 eingeführten symbolischen und stochastischen Parsingverfahren verallgemeinert. Es wird die mathematische Struktur eines *Semirings*, sowie *CKY-*, *Count-*, *Inside-*, *Viterbi-* und *Parsewald-Semiring* eingeführt. Drei weitere stochastische Parsingverfahren werden über ihre Semiringe eingeführt: *Viterbi-Parse-*, *N-Best-*, und *N-Best-Parse-Semiring*. Im Vergleich zu Goodman (1998) werden neue(!), einfache und genaue Definitionen benutzt. Es wird gezeigt, dass der Semiring-Parsing-Algorithmus dynamische Programmierung benutzt. Es wird ferner gezeigt, dass der Semiring-Parsing-Algorithmus effizient ist, wenn die Semiring-Operationen in konstanter Zeit vorgenommen werden können.

In Abschnitt 2.8 werden die drei wichtigsten Sprachmodelle eingeführt: *n-gram-Modelle*, sowie *Markov-* und *Hidden Markov-Modelle*. Es fällt auf, dass n-gram-Modelle weniger allgemein als Markov-Modelle sind, und diese wiederum weniger allgemein als Hidden Markov-Modelle. Es wird gezeigt, dass alle drei Sprachmodelle durch probabilistische kontextfreie Grammatiken simuliert werden können und es wird darauf hingewiesen, dass der berühmte *Forward-Backward-Algorithmus* (Baum 1972) ein einfacher Spezialfall innerhalb der EM-Theorie ist.

Zur Vertiefung in die (linguistische) Stochastik wird die folgende Literatur empfohlen: An deutsch-sprachigen Büchern sind die gleichwertigen kurzen Einführungen von Krengel (1987) oder Bosch (1995) und Bosch (1997) zu nennen. Einen umfassenden Einblick ermöglicht Bauer (1978), der insbesondere auch die Masstheorie behandelt. Einen schnellen Einstieg bietet Aigner (1996), wobei die relevanten Informationen zwar über das ganze Buch verstreut sind, dafür aber insbesondere diskrete Wahrscheinlichkeitsverteilungen besprochen werden. Hervorragend ist Dotzauer (1987), in dem verraten wird, wie Wahrscheinlichkeitsverteilungen effizient simuliert werden. An englisch-sprachiger Literatur ist vor allem DeGroot (1989) zu nennen. Er bietet eine hervorragende Einführung mit vielen Beispielen. Cover und Thomas (1991) müssen gelesen werden. Der Schwerpunkt dieses Standardwerkes liegt auf diskreten Wahrscheinlichkeitsverteilungen innerhalb der Informationstheorie. Speziell für den Linguisten ist die auf dem Internet verfügbare Einführung von Krenn und Samuelsson (1997) empfehlenswert. Schnell schlau macht man sich mit

Manning und Schütze (1999). Sehr hilfreich ist der umfangreiche Literaturteil. Das neueste Buch ist von Jurawsky und Martin (2000). Das Standardwerk ist aber weiterhin der schmale Band des bedeutensten Pioniers der stochastischen Linguistik, Charniak (1993).

An weiteren mathematischen Kenntnissen werden Grundlagen-Kenntnisse der *Differentialrechnung mehrerer Veränderlichen*, sowie der *linearen Algebra* benötigt. Die aus beiden Gebieten benutzten Definitionen und Sätze werden bis auf eine Ausnahme in diesem Kapitel präsentiert. Bei dieser Ausnahme handelt es sich um den Begriff der *(partiellen) Ableitung*. Leider ist er von zentraler Bedeutung, weil Abschnitt 2.4 ohne Fertigkeiten im Umgang mit (partiellen) Ableitungen nicht verstanden werden kann. Die zwei sehr guten Bücher von Erwe (1962) und die schmalen Bände von Forster (1981) verschaffen Abhilfe: sie führen in die *Analysis* ein. Das kleine Buch von Fischer (1980) hilft bei der *linearen Algebra*.

2.1 Wahrscheinlichkeiten, Häufigkeiten, Korpora

Das folgende Zitat ist von C.E. Shannon (1949), der das mathematische Fundament zur modernen Informationstheorie legte:

> The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have *meaning*; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem. The significant aspect is that the actual message is one *selected from a set* of possible messages. The system must be designed to operate for each possible selection, not just the one which will actually be chosen since this is unknown at the time of design. If the number of messages is finite then this number (...) can be regarded as a measure of information produced when one message is chosen from the set, all choices being equally likely.

In diesem Zitat kommen zwei für einen Linguisten sicherlich überraschende Standpunkte zum Ausdruck: (i) Shannon sieht sprachliche Äusserungen ("messages") lediglich als Elemente einer (endlichen) Menge von möglichen sprachlichen Äusserungen an. (ii) Ihn scheint an einer sprachlichen Äusserung zunächst nur die Information zu interessieren, mit welcher Wahrscheinlichkeit sie aus dieser Menge ausgewählt oder generiert wird. Auch für die stochastische Grundlage der Linguistik sind diese zwei Aspekte sprachlicher Äusserungen von fundamentaler Bedeutung, da der erste Aspekt die Einführung einer Menge von linguistischen Datentypen motiviert, und der zweite Aspekt zum zentralen Begriff der Wahrscheinlichkeitsverteilung selbst führt. Da Wahrscheinlichkeitsverteilungen auf abzählbaren Mengen[1] mit besonders einfachen mathematischen Mitteln eingeführt werden können, wird angenommen, dass jede linguistisch interessante Menge $\mathcal{Y}$ abzählbar ist. Diese Annahme kann für die meisten Mengen von linguistischen Datentypen, die im weiteren Verlauf dieser Arbeit auftauchen werden, leicht verifiziert werden:

- die endliche Menge deutscher/englischer *Buchstaben*
- eine endliche Teilmenge der deutschen/englischen *Wörter*
- die unendliche Menge deutscher/englischer *Texte*[2]

[1] *Abzählbare Mengen* sind entweder endlich oder bijektiv auf die Menge der natürlichen Zahlen $\mathcal{N}$ abbildbar.

[2] Ein *Text* ist eine endliche Folge von Buchstaben oder Wörtern (incl. Sonderzeichen, wie zum Beispiel Leerzeichen und Interpunktion).

- die endliche Menge deutscher/englischer *Vokale* und *Diphtonge*
- eine endliche Teilmenge der deutschen/englischen *Konsonatencluster*
- eine endliche Teilmenge der deutschen/englischen *Silben*
- eine endliche Teilmenge der deutschen/ englischen *klassifizierten Silben*
- eine endliche Teilmenge der deutschen/englischen *Adjektive* und *mit Subkategorisierungsrahmen versehenen Verben*
- eine endliche Teilmenge der deutschen/englischen *Nomen*
- eine endliche Teilmenge der deutschen/englischen *subkategorisierten Verb-Nomen-Paare*
- eine endliche Teilmenge der deutschen/englischen *klassifizierten und subkategorisierten Verb-Nomen-Paare*
- die unendliche Menge aller *Sätze einer (lexikalisierten) kontextfreien deutschen Grammatik*
- die unendliche Menge aller *Syntaxbäume einer (lexikalisierten) kontextfreien deutschen Grammatik*
- eine endliche Teilmenge der *Sätze einer unifikationsbasierten deutschen Grammatik*
- eine endliche Teilmenge[3] der *mit Feature-Strukturen versehenen Syntaxbäume einer unifikationsbasierten deutschen Grammatik*

Eine *Wahrscheinlichkeitsverteilung* $p(.)$ auf $\mathcal{Y}$ ist eine Funktion mit den linguistischen Datentypen $\mathcal{Y}$ als Definitionsbereich und dem Intervall $[0,1]$ als Wertebereich:

$$p : \mathcal{Y} \rightarrow [0,1]$$

sodass gilt:

$$\sum_{y \in \mathcal{Y}} p(y) = 1 \ . \tag{2.1}$$

Eine Wahrscheinlichkeitsverteilung $p(.)$ weist jedem linguistischen Datentyp $y \in \mathcal{Y}$ eine *Wahrscheinlichkeit*, d.h. eine Zahl $p(y)$ zwischen Null und Eins zu. Der *sichere Datentyp* $y \in \mathcal{Y}$ hat hierbei die maximale Wahrscheinlichkeit $p(y) = 1$, während ein *unmöglicher*

[3]Die Wahrscheinlichkeitsverteilung auf dieser endlichen Teilmenge aller Grammatikanalysen kann sehr einfach zu einem Mass auf der vollständigen Menge aller Grammatikanalysen fortgesetzt werden und dann ohne Schwierigkeiten zur Disambiguierung benutzt werden;

( ) , . ; Frequently If The These a according actual actually all another approximately are as aspect aspects at be being can certain choices chosen communication conceptual correlated design designed each either engineering entities equally exactly finite for from fundamental have information irrelevant is just likely meaning measure message messages must not number of one operate or physical point possible problem produced refer regarded reproducing selected selection semantic set significant since some system that the then they this time to unknown when which will with

Abbildung 2.1: Die 87 Datentypen aus Shannons Zitat in alphabetischer Reihenfolge.

. (9) the (7) of (7) is (7) to (4) one (4) that (3) or (3) messages (3) message (3) be (3) at (3) a (3) The (3) this (2) system (2) set (2) selected (2) problem (2) possible (2) point (2) number (2) from (2) communication (2) chosen (2) are (2) , (2) with (1) will (1) which (1) when (1) unknown (1) time (1) they (1) then (1) some (1) since (1) significant (1) semantic (1) selection (1) reproducing (1) regarded (1) refer (1) produced (1) physical (1) operate (1) not (1) must (1) measure (1) meaning (1) likely (1) just (1) irrelevant (1) information (1) have (1) fundamental (1) for (1) finite (1) exactly (1) equally (1) entities (1) engineering (1) either (1) each (1) designed (1) design (1) correlated (1) conceptual (1) choices (1) certain (1) can (1) being (1) aspects (1) aspect (1) as (1) approximately (1) another (1) all (1) actually (1) actual (1) according (1) These (1) If (1) Frequently (1) ; (1)) (1) ((1)

Abbildung 2.2: Die Vorkommenshäufigkeiten der Datentypen in Shannons Zitat (sortiert nach Häufigkeiten).

Datentyp $y' \in \mathcal{Y}$ die minimale Wahrscheinlichkeit $p(y') = 0$ besitzt. Die Gleichung (2.1) besagt, dass die Summe der Wahrscheinlichkeiten aller Datentypen genau Eins ergibt[4]. Dies bedeutet, dass es einen sicheren Datentyp nur dann geben kann, wenn alle anderen Datentypen unmögliche Datentypen sind. Der Sinn einer Wahrscheinlichkeitsverteilung besteht daher darin, dass sie für jeden Datentyp die Information bereit hält, mit welcher Wahrscheinlichkeit dieser Datentyp aus einer Grundmenge von Datentypen heraus erzeugt wird. Mit diesem Aspekt ist die Frage sehr eng verbunden, wie oft ein linguistischer Datentyp in einem gegebenem Korpus vorkommt.

Im folgenden wird das Zitat von Shannon als Beispiel benutzt. Wenn die Satzzeichen (Punkt, Komma, Semikolon, runde Klammern) als Wörter gelten, besteht Shannons Zitat

[4] Aus mathematischer Sicht ist noch erwähnenswert, dass in Gleichung (2.1) die Summationsreihenfolge der (evtl. unendlichen) Reihe $\sum_{y \in \mathcal{Y}} p(y)$ nicht festgelegt ist. Ein allgemeiner Satz der Analysis besagt aber, dass eine *absolut konvergente Reihe* umgeordnet werden darf (Erwe 1962). Da alle Summanden $p(y)$ positiv sind, liegt dieser Fall in Gleichung (2.1) glücklicherweise vor.

. (0.06081) the (0.04729) of (0.04729) is (0.04729) to (0.02702) one (0.02702) that (0.02027) or (0.02027) messages (0.02027) message (0.02027) be (0.02027) at (0.02027) a (0.02027) The (0.02027) this (0.01351) system (0.01351) set (0.01351) selected (0.01351) problem (0.01351) possible (0.01351) point (0.01351) number (0.01351) from (0.01351) communication (0.01351) chosen (0.01351) are (0.01351) , (0.01351) with (0.00675) will (0.00675) which (0.00675) when (0.00675) unknown (0.00675) time (0.00675) they (0.00675) then (0.00675) some (0.00675) since (0.00675) significant (0.00675) semantic (0.00675) selection (0.00675) reproducing (0.00675) regarded (0.00675) refer (0.00675) produced (0.00675) physical (0.00675) operate (0.00675) not (0.00675) must (0.00675) measure (0.00675) meaning (0.00675) likely (0.00675) just (0.00675) irrelevant (0.00675) information (0.00675) have (0.00675) fundamental (0.00675) for (0.00675) finite (0.00675) exactly (0.00675) equally (0.00675) entities (0.00675) engineering (0.00675) either (0.00675) each (0.00675) designed (0.00675) design (0.00675) correlated (0.00675) conceptual (0.00675) choices (0.00675) certain (0.00675) can (0.00675) being (0.00675) aspects (0.00675) aspect (0.00675) as (0.00675) approximately (0.00675) another (0.00675) all (0.00675) actually (0.00675) actual (0.00675) according (0.00675) These (0.00675) If (0.00675) Frequently (0.00675) ; (0.00675)) (0.00675) ((0.00675)

Abbildung 2.3: Die empirischen Wahrscheinlichkeiten der Datentypen in Shannons Zitat (sortiert nach Wahrscheinlichkeiten).

aus 148 englischen Wörtern, wovon 87 dieser Wörter (etwas mehr als die Hälfte) verschieden sind. Diese Menge ist natürlich eine Menge linguistischer Datentypen. Ihre Elemente werden in alphabetischer Reihenfolge in der Abbildung 2.1 gezeigt[5]. In Abbildung 2.2 werden die Wörter aus Shannons Zitat zusammen mit ihren Vorkommenshäufigkeiten gezeigt[6]. Es ist bemerkenswert, dass am Anfang dieser Liste nur wenige *Inhaltswörter* dafür aber sehr viele *Funktionswörter* vorkommen. Schön ist immerhin, dass sich *message* und *messages* unter den 14 häufigsten Wörtern des Zitates befinden. Wenn man die Häufigkeit eines Wortes durch die Zahl 148 (der Zahl[7] aller Wörter in Shannons Zitat) teilt, erhält man die in Abbildung 2.3 in Klammern gezeigten empirischen Wahrscheinlichkeiten. Die Summe aller dieser Werte ist 1, sodass sie eine Wahrscheinlichkeitsverteilung auf der Menge aller im Zitat gebrauchten Wörter bilden. Von dieser Wahrscheinlichkeitsverteilung könnte man behaupten, dass sie Shannons "Wortwahl" am Anfang seines berühmten Artikels näherungsweise widergibt.

Shannons Zitat ist ein sehr kleiner *Korpus*. Der Begriff Korpus ist für die stochastische Linguistik ein fundamentaler Begriff und wird wie folgt definiert:

Eine endliche Folge von Datentypen $< y_1, \ldots, y_n > \in \mathcal{Y}^n$ wird *Korpus der Länge n* genannt. Die im Korpus vorkommenden Datentypen können natürlich leicht als die Menge $\{y_1, \ldots, y_n\}$ notiert werden. Beachtenswert ist jedoch, dass diese Menge nicht genau n Elemente enthalten muss, da einige Datentypen mehrfach im Korpus vorhanden sein können. Die *Anzahl der Datentypen im Korpus* ist somit die Kardinalität:

$$\#\{y_1, \ldots, y_n\} \ .$$

Da es wünschenswert erscheint, von einem Datentyp im Korpus nicht nur sagen zu können, ob er im Korpus (mehrfach) vorkommt, sondern auch an welchen Korpuspositionen, wird der Begriff des Datentokens eingeführt:

[5] Diese Liste kann auf einer *UNIX-Shell* sehr einfach erzeugt werden. Ist Shannons Zitat in der Textdatei **shannon.txt** abgespeichert, und hat man ein Programm **tokenize** zur Hand, welches eine Textdatei in eine Liste von Wörtern konvertiert, so leistet der folgenden Befehl das Gewünschte:

```
cat shannon.txt | tokenize | sort -u
```

Steht **tokenize** nicht zur Verfügung, so leistet der folgende Befehl relativ gute Dienste:

```
alias tokenize = " tr ' ' '\n' "
```

[6] Auch diese Liste kann auf einer *UNIX-Shell* einfach erzeugt werden:

```
cat shannon.txt | tokenize | sort | uniq -c | sort -n -r
```

[7] Sie kann auf einer *UNIX-Shell* so erzeugt werden:

```
cat shannon.txt | tokenize | wc -l
```

ein Paar $<y, i>$ aus einem Datentyp $y \in \mathcal{Y}$ und einer *Korpusposition* $i \in \{1, \ldots, n\}$ wird *Datentoken* genannt, falls $y = y_i$.

Gemäss dieser Definition kann ein Datentyp nie ein Datentoken sein (selbst wenn der Datentyp im Korpus vorkommt). Ein Datentyp ist erst dann ein Datentoken, wenn er zusammen mit seiner Korpusposition, an der dieser Datentyp im Korpus erscheint, auftritt. Die Menge der im Korpus vorkommenden Datentoken $\{<y_1, 1>, \ldots, <y_n, n>\}$ lässt sich mit dem Korpus $<y_1, \ldots, y_n>$ selbst identifizieren und hat genau n Elemente. Natürlich liegt es auf der Hand, einem Datentoken $<y, i>$ die Wahrscheinlichkeit seines Datentyps y zuzuordnen:

$$p(<y, i>) = p(y) \ .$$

Diese Zuordnung erzeugt jedoch im allgemeinen keine Wahrscheinlichkeitsverteilung auf der Menge der Datentoken eines festen Korpus . Trotzdem ist die angegebene Definition der Datentoken-Wahrscheinlichkeit gerechtfertigt, da man mit ihr offensichtlich eine Wahrscheinlichkeitsverteilung $p(.)$ auf der Menge aller Datentoken erhält, die an einer festen Korpusposition i vorkommen können, wenn man die Menge aller Korpora durchläuft:

$$p : \mathcal{Y} \times \{i\} \to [0, 1]$$

Es ist klar, dass diese Wahrscheinlichkeitsverteilung gemäss Definition nicht von der fixierten Korpusposition i abhängig ist. Zusammenfassend besteht der wesentliche Aspekt des Begriffs des Datentokens darin, einen Datentyp mittels der Korpusposition in einem Korpus zu lokalisieren. Die Länge des Shannon Zitats ist 148. Die Anzahl seiner Datentypen ist 87 und die Anzahl der Datentoken im Korpus ist 148. Die Korpuspositionen sind $\{1 \ldots 148\}$. Datentoken sind zum Beispiel: $< \mathit{The}, 1 >$, $< \mathit{of}, 4 >$, $< \mathit{of}, 8 >$, oder $< ., 148 >$. Das Paar $< \mathit{messages}, 12 >$ ist kein Datentoken, da "messages" nicht an Position 12 im Zitat auftaucht.

Der zentrale Begriff der *Korpuswahrscheinlichkeit* wird als Produkt der Wahrscheinlichkeiten aller im Korpus vorkommenden Datentoken eingeführt:

$$p(y_1, \ldots, y_n) = \prod_{i=1}^{n} p(<y_i, i>) = \prod_{i=1}^{n} p(y_i) = \prod_{y=y_1}^{y_n} p(y) \ .$$

In dieser Definition, die aus mathematischer Sicht die *stochastische Unabhängigkeit der Datentoken im Korpus* ausdrückt, geht die Wahrscheinlichkeit jedes Datentyps im Korpus genauso oft ein, wie dieser Datentyp im Korpus vorkommt. Im allgemeinen gilt:

$$\prod_{y=y_1}^{y_n} p(y) \neq \prod_{y \in \{y_1, \ldots, y_n\}} p(y) \ .$$

Die Aussage, dass die Korpuswahrscheinlichkeit das Produkt der Wahrscheinlichkeiten aller im Korpus vorkommenden Datentypen ist nicht korrekt. Wenn man dies irgendwo liest,

sollte man sich bewusst sein, dass der Verfasser sich ungenau ausgedrückt hat und implizit die Wahrscheinlichkeiten aller im Korpus vorkommenden Datentypen *zusammen* mit ihren Häufigkeiten meint. Die folgende Formel beweist, dass durch die Definition der Korpuswahrscheinlichkeit auch tatsächlich eine Wahrscheinlichkeitsverteilung auf $\mathcal{Y}^n$ gegeben ist, d.h. auf der Menge der Korpora der festen Korpuslänge n. Dabei wird ausgenutzt, dass $p(.)$ eine Wahrscheinlichkeitsverteilung auf $\mathcal{Y}$ ist, d.h. für alle Korpuspositionen $i \in \{1, \ldots, n\}$ die Gleichung $\sum_{y_i \in \mathcal{Y}} p(y_i) = 1$ gilt:

$$\begin{aligned}
\sum_{\langle y_1,\ldots,y_n\rangle \in \mathcal{Y}^n} p(y_1, \ldots, y_n) &= \sum_{\langle y_1,\ldots,y_n\rangle \in \mathcal{Y}^n} \prod_{i=1}^{n} p(y_i) \\
&= \sum_{y_1 \in \mathcal{Y}} \cdots \sum_{y_{n-1} \in \mathcal{Y}} \sum_{y_n \in \mathcal{Y}} p(y_1) \cdots p(y_{n-1}) \cdot p(y_n) \\
&= \sum_{y_1 \in \mathcal{Y}} p(y_1) \cdots \sum_{y_{n-1} \in \mathcal{Y}} p(y_{n-1}) \cdot \sum_{y_n \in \mathcal{Y}} p(y_n) \\
&= \sum_{y_1 \in \mathcal{Y}} p(y_1) \cdots \sum_{y_{n-1} \in \mathcal{Y}} p(y_{n-1}) \cdot 1 \\
&= \sum_{y_1 \in \mathcal{Y}} p(y_1) \cdots \sum_{y_{n-1} \in \mathcal{Y}} p(y_{n-1}) \\
&\vdots \\
&= 1 \; .
\end{aligned}$$

Die Korpuswahrscheinlichkeit ist wegen der Kommutativität der Multiplikation reeller Zahlen gegen Vertauschungen innerhalb der Reihenfolge der Datentypen im Korpus robust. Daher ist man in der Regel lediglich an der Häufigkeitsverteilung aller Datentypen im Korpus interessiert, d.h. an der Information, welche linguistischen Datentypen in dem Korpus wie oft vorkommen. Eine *Häufigkeitsverteilung* $f(.)$ ist eine Funktion mit den Datentypen $\mathcal{Y}$ als Definitionsbereich und den *natürlichen Zahlen* als Wertebereich:

$$f : \mathcal{Y} \to \mathcal{N}$$

wobei die *Häufigkeit* eines Datentyps $y \in \mathcal{Y}$ durch:

$$f(y) = \#\{i \in \mathcal{N} \mid y_i = y\} \; .$$

gegeben ist. Die *absolute Häufigkeit* $|f|$ ist die Summe der Häufigkeiten aller Datentypen im Korpus:

$$|f| = \sum_{y \in \mathcal{Y}} f(y) \; . \tag{2.2}$$

Natürlich ist die absolute Häufigkeit mit der Korpuslänge identisch: $|f| = n$. Damit kann für:

$$p(f) = \prod_{y \in \mathcal{Y}} p(y)^{f(y)} \tag{2.3}$$

die Übereinstimmung mit der Korpuswahrscheinlichkeit leicht nachgerechnet werden:

$$p(f) = p(y_1, \ldots, y_n) \ .$$

Diese Identität bestätigt, dass die Korpuswahrscheinlichkeit nur von der Häufigkeitsverteilung aller Datentypen im Korpus abhängt, und nicht von der Reihenfolge der Datentypen im Korpus. Deshalb kann ein Korpus $<y_1, \ldots, y_n>$ mit seiner Häufigkeitsverteilung $f(.)$ identifiziert werden. Die *empirische Wahrscheinlichkeitsverteilung* $\tilde{p}(.)$ hebt den formalen Nachteil der Häufigkeitsverteilung aller Datentypen im Korpus, keine Wahrscheinlichkeitsverteilung zu sein, auf:

$$\tilde{p} : \mathcal{Y} \rightarrow [0, 1]$$

Einem Datentyp $y \in \mathcal{Y}$ wird dabei die folgende *empirische Wahrscheinlichkeit* zugewiesen:

$$\tilde{p}(y) = |f|^{-1} \cdot f(y) \tag{2.4}$$

Die Gleichung (2.2) weist darauf hin, dass $\tilde{p}(.)$ tatsächlich eine Wahrscheinlichkeitsverteilung auf $\mathcal{Y}$ ist. Abbildung 2.3 zeigt die empirische Wahrscheinlichkeitsverteilung für Shannons Zitat. In dieser Arbeit wird immer wieder verwendet, dass ein Korpus mit seiner Häufigkeitsverteilung (oder der empirischen Wahrscheinlichkeitsverteilung) "identifiziert" werden kann. Natürlich geht mit dieser Identifizierung ein *Informationsverlust* einher: Zum Beispiel wird auf Wortebene anstatt mit Shannons Originalzitat, mit der in Abbildung 2.2 gezeigten Häufigkeitsverteilung gearbeitet. Aus dieser Häufigkeitsverteilung sind jedoch die Sätze des Originalzitats nicht mehr rekonstruierbar. Ein Informationsverlust auf und oberhalb der Syntaxebene ist eingetreten. Auf Satzebene sind Sätze die linguistischen Datentypen. Auch die Häufigkeitsverteilung der Sätze enthält weniger Information als Shannons Originalzitat. Weil die genaue Abfolge der Sätze nicht mehr rekonstruierbar ist, tritt der Informationsverlust hier auf und oberhalb der Diskursebene ein. Die Aussage, dass ein Korpus mit seiner Häufigkeitsverteilung identifiziert werden kann, ist also streng genommen nicht richtig: Je nach verwendetem linguistischen Datentyp tritt bei der Identifikation von Korpus und Häufigkeitsverteilung ein kleiner oder grosser Informationsverlust auf. Der wichtige Punkt ist, dass in der linguistischen Stochastik dieser Informationsverlust in Kauf genommen wird. Der einzige Grund dafür ist, dass sich die Korpuswahrscheinlichkeit, wie oben gezeigt wurde, aus der Häufigkeitsverteilung $f(.)$ und der gegebenen Wahrscheinlichkeitsverteilung $p(.)$ berechnen lässt. Da sich in den nächsten Abschnitten erweisen wird, dass die Korpuswahrscheinlichkeit der zentrale Begriff der linguistischen Stochastik ist, scheint die Identifikation von Korpus und Häufigkeitsverteilung gerechtfertigt zu sein.

2.2 Ereignisse und konsistente kontextfreie Grammatiken

Unifikationsbasierte Grammatikformalismen (Johnson 1988), wie LFG oder HPSG, wurden in den letzten zehn Jahren zu den wichtigsten Grammatikansätzen in der Computerlinguistik und verdrängten einfachere Ansätze wie Finite-State-Technologie (AT&T) oder kontextfreie Grammatiken aus der Praxis, die noch in den 80er Jahren die Szene beherrschten. Parallel zu dieser Entwicklung gewann die stochastische Linguistik innerhalb der Computerlinguistik an Bedeutung. Ein früher Versuch im Rahmen der stochastischen Linguistik, eine unifikationsbasierte Grammatik mit einem Wahrscheinlichkeitsmodell zu versehen, ist die Arbeit von Briscoe und Carroll (1993); die neuesten Arbeiten vielleicht von Bod und Kaplan (1998), Schmid (1999b), Johnson et al. (1999), Johnson und Riezler (2000), Bod (2000) und Riezler et al. (2000). Leider mangelt es allen diesen Ansätzen an einem effizienten Satz-Disambiguierungsverfahren. Obwohl beispielsweise in der Arbeit von Riezler et al. (2000) ein Trainingsverfahren für eine stochastische unifikationsbasierte Grammatik vorgestellt wird, und die dort vorgenommene Evaluierung sehr gute Ergebnisse für niedrig-ambige Sätze zeigt, ist eine Anwendung dieses Modells auf hoch-ambige Sätze, wie sie normalerweise in freiem Text vorkommen, praktisch undurchführbar (und wurde in den Evaluierungen nicht vorgenommen).

Im Gegensatz hierzu ist es mit probabilistischen kontextfreien Grammatiken mit dem sogenannten *Inside-Algorithmus* möglich, für jeden Satz aus der Sprache einer kontextfreien Grammatik *effizient* zu berechnen, mit welcher Wahrscheinlichkeit dieser Satz, in der durch die Grammatik beschriebenen Sprache, vorkommt. Die wesentliche Eigenschaft von probabilistischen kontextfreien Grammatiken ist jedoch, dass mit dem *Viterbi-Parse-Algorithmus* (oder mit dem *N-Best-Parse-Algorithmus*) für einen Satz seine wahrscheinlichste Analyse (oder die Analysen der N-besten Wahrscheinlichkeiten) *effizient* berechenbar ist. Natürlich wurde diese Eigenschaft probabilistischer kontextfreier Grammatiken in vielen Anwendungen benutzt; in neuerer Zeit zum Beispiel in Charniak (1997), Carroll und Rooth (1998), Beil et al. (1999), Charniak (2000), Schulte im Walde und Schmid (2000), um die Analysen eines grammatischen Satzes zu disambiguieren, indem die wahrscheinlichste Analyse als die "beste" oder "zutreffendste" Analyse dieses Satzes angesehen wird.

Solange keine effizienten Disambiguierungsalgorithmen für stochastische unifikationsbasierte Grammatiken gefunden werden, werden reguläre oder kontextfreie Grammatiken innerhalb der angewandten stochastischen Linguistik weiterhin eine herausragende Stellung einnehmen. Ein sehr interessanter Ansatz, effiziente Disambiguierungsalgorithmen für stochastische unifikationsbasierte Grammatiken zu finden, besteht darin, im Rahmen der

Optimalitätstheorie zunächst zu versuchen, die Anzahl der Analysen symbolisch stark zu beschränken (Frank et al. (1998), Kuhn (2000)), und anschliessend diese wenigen Analysen nacheinander stochastisch zu bewerten. Der Nachteil einer solchen "2-Level-Syntax" ist leider, dass das Gesamtsystem höchstens so gut sein kann, wie das schwächere der beiden Teilsysteme. Ferner ist das Effizienzproblem nicht wirklich gelöst, sondern auf die Ebene der Symbolik verschoben worden.

Im weiteren Verlauf dieses Abschnitts werden probabilistische kontextfreie Grammatiken eingeführt, für welche in Abschnitt 2.5 und 3.1 effiziente Trainingsverfahren, und in Abschnitt 2.6 und 2.7 effiziente Disambiguierungsverfahren vorgestellt werden. Im folgenden wird gezeigt werden, dass die Menge der Analysen eines Satzes ein *linguistisches Ereignis* ist. Deshalb ist es vorteilhaft, den in Abschnitt 2.1 für eine Menge von linguistischen Datentypen vorgestellten Begriff der Wahrscheinlichkeitsverteilung auf die Menge der linguistischen Ereignisse zu erweitern. Ausserdem liegt mit dieser Verallgemeinerung der Begriff der Wahrscheinlichkeitsverteilung in seiner bekannten axiomatischen Gestalt vor:

Ist $\mathcal{X}$ eine Menge linguistischer Datentypen, so wird jede Teilmenge $X \subseteq \mathcal{X}$ als *linguistisches Ereignis* bezeichnet. Die Standard-Interpretation ist, dass X das Ereignis des Eintreffens von mindestens einem der alternativen Datentypen $x \in X$ ist. Die Menge aller linguistischen Ereignisse, d.h. die Potenzmenge von $\mathcal{X}$ bezeichnet man mit dem Symbol $\mathcal{P}(\mathcal{X})$. Eine Wahrscheinlichkeitsverteilung auf $\mathcal{X}$, so wie sie in Abschnitt 2.1 eingeführt wurde,

$$p : \mathcal{X} \rightarrow [0,1]$$

mit

$$\sum_{x \in \mathcal{X}} p(x) = 1 \ ,$$

kann sehr einfach *mathematisch fortgesetzt* werden, sodass eine *Wahrscheinlichkeitsverteilung auf der Menge der linguistischen Ereignisse* entsteht:

$$p : \mathcal{P}(\mathcal{X}) \rightarrow [0,1] \ .$$

Dazu wird die *Wahrscheinlichkeit eines linguistischen Ereignisses* $X \in \mathcal{P}(\mathcal{X})$ als Summe der Wahrscheinlichkeiten aller Datentypen $x \in X$ definiert:

$$p(X) := \sum_{x \in X} p(x) \ .$$

Die Wahrscheinlichkeit $p(X)$ kann als die Wahrscheinlichkeit interpretiert werden, dass mindestens eines der Datentypen aus X eintrifft. Man kann sehr einfach nachprüfen, dass

$S \rightarrow NP\ schläft$	(1.0)
$S \rightarrow Hans$	(0.7)
$NP \rightarrow Hans$	(0.3)

Abbildung 2.4: Probabilistische kontextfreie Grammatik mit Verletzung der Standard-Nebenbedingung für die Grammatikregel-Wahrscheinlichkeiten.

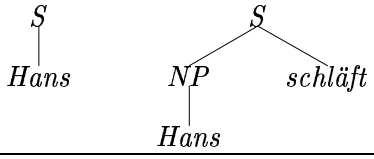


Abbildung 2.5: Die Menge aller Syntaxbäume der probabilistischen kontextfreien Grammatik aus Abbildung 2.4.

auf diese Weise die drei folgenden *Wahrscheinlichkeitsaxiome* erfüllt werden:

(*Nicht-Negativität*)	$p(X) \geq 0$	für $X \in \mathcal{P}(\mathcal{X})$
(*Konsistenz*)	$p(\mathcal{X}) = 1$	
(*Additivität*)	$p(X + X') = p(X) + p(X')$	für disjunkte $X,\ X' \in \mathcal{P}(\mathcal{X})$

Interessant ist hierbei vor allem das dritte Axiom: $X + X'$ ist die Vereinigung der beiden disjunkten Mengen X und X' und kann als Ereignis des Eintreffens mindestens eines der beiden Ereignisse X oder X' interpretiert werden. Das Axiom besagt in Übereinstimmung mit der Intuition, dass die Wahrscheinlichkeit dieses Ereignisses sich additiv aus den Einzelwahrscheinlichkeiten berechnet.

Mit Hilfe des Begriffs des Ereignisses und der Wahrscheinlichkeit eines Ereignisses werden nun probabilistische kontextfreie Grammatiken eingeführt. Hierfür wird Vertrautheit mit dem Begriff der kontextfreien Grammatik vorausgesetzt. Eine *probabilistische kontextfreie Grammatik* ist eine *kontextfreie Grammatik*, wobei jeder Grammatikregel eine Wahrscheinlichkeit zugeordnet ist. Abbildung 2.4 zeigt eine sehr einfache probabilistische kontextfreie Grammatik, bestehend aus drei Grammatikregeln. In der ersten Spalte der Abbildung sind die Grammatikregeln angegeben, in der zweiten, die ihnen zugeordnete Wahrscheinlichkeit (in runden Klammern). Da die Summe aller Grammatikregel-Wahrscheinlichkeiten in Abbildung 2.4 den Wert 2.0 ergibt, liegt bei dieser und allen anderen probabilistischen kontextfreien Grammatiken keine Wahrscheinlichkeitsverteilung über die Menge der Grammatikregeln vor. Es ist daher schwer zu verstehen, warum diese Werte als Wahrscheinlichkeiten bezeichnet werden. Im folgenden wird versucht, eine Lösung für dieses Problem zu finden.

Der Schlüsselschritt zu einem besseren Verständnis ist, dass die Grammatikregel-Wahr-

scheinlichkeiten dazu benutzt werden können, den Syntaxbäumen der probabilistischen kontextfreien Grammatik eine Wahrscheinlichkeit zuzuweisen:

Für eine Grammatikregel $r \in G$ wird ihre linke Seite mit $\mathrm{lhs}(r)$ und ihre Wahrscheinlichkeit mit $p(r)$ notiert. Ist $\mathcal{T}(G)$ die Menge aller Syntaxbäume der Grammatik G, so wird mit $f_r(x)$ die Häufigkeit einer Grammatikregel $r \in G$ in einem Syntaxbaum $x \in \mathcal{T}(G)$ notiert. Für $f_r(x) > 0$, schreibt man $r \in x$, anderenfalls $r \notin x$. Mit dieser Notation ist die Wahrscheinlichkeit eines Syntaxbaumes $x \in \mathcal{T}(G)$ folgendermassen definiert:

$$p(x) = \prod_{r \in G} p(r)^{f_r(x)} = \prod_{r \in x} p(r)^{f_r(x)}$$

Im folgenden wird diese Definition anhand der Beispielsgrammatik aus Abbildung 2.4 vorgestellt. Die Menge aller möglichen Syntaxbäume $\mathcal{T}(G)$ ist in Abbildung 2.4 dargestellt. Der linke Syntaxbaum x_L hat die Wahrscheinlichkeit

$$p(x_L) = \prod_{r \in x_L} p(r)^{f_r(x_L)} = p(S \rightarrow \mathit{Hans}) = 0.7 \ ,$$

der rechte Syntaxbaum x_R hat die Wahrscheinlichkeit

$$p(x_R) = \prod_{r \in x_R} p(r)^{f_r(x_R)} = p(S \rightarrow \mathit{Hans}\ \ VP) \cdot p(VP \rightarrow \mathit{schläft}) = 1.0 \cdot 0.3 = 0.3 \ .$$

Da die Summe dieser beiden Wahrscheinlichkeiten 1.0 ergibt, induzieren die Grammatikregel-Wahrscheinlichkeiten für dieses Beispiel eine Wahrscheinlichkeitsverteilung auf ganz $\mathcal{T}(G)$.

Aus der Definition der Syntaxbaum-Wahrscheinlichkeit lässt sich mit dem Begriff des Ereignisses die Definition für die Wahrscheinlichkeit eines Satzes aus der Sprache $\mathcal{L}(G)$ ableiten. Der Schlüsselschritt ist, einen Satz mit der Menge aller seiner Syntaxbäume zu identifizieren: Bezeichnet $\mathcal{T}(y)$ die Menge aller Syntaxbäume eines Satzes $y \in \mathcal{L}(G)$, so ist die Wahrscheinlichkeit von $y \in \mathcal{L}(G)$ wie folgt definiert:

$$p(y) = \sum_{x \in \mathcal{T}(y)} p(x) \ .$$

Die Interpretation dieser Definition ist im Kontext der linguistischen Ereignisse daher: ein Satz $y \in \mathcal{L}(G)$ tritt genau dann als Ereignis ein, wenn mindestens einer seiner Analysen eintritt. In Übereinstimmung mit dieser Interpretation ist die Wahrscheinlichkeit eines Satzes die Summe der Wahrscheinlichkeiten aller seiner Analysen. Die Sprache der Beispielgrammatik aus Abbildung 2.4 besteht nur aus zwei grammatischen Sätzen:

$$\mathcal{L}(G) = \{\mathit{Hans},\ \mathit{Hans\ schläft}\} \ .$$

Da beide Sätze unambig sind, stimmen ihre Wahrscheinlichkeiten mit den Wahrscheinlichkeiten ihrer Analysen überein.

Es ist interessant, dass die Grammatikregel-Wahrscheinlichkeiten der Beispielgrammatik so gewählt sind, dass hieraus eine Wahrscheinlichkeitsverteilung auf der Menge $\mathcal{T}(G)$ aller Syntaxbäume resultiert. Dies ist die Anforderung, die man an jede probabilistische kontextfreie Grammatik G stellen wird: ***Die Grammatikregel-Wahrscheinlichkeit müssen so gewählt sein, dass sie eine Wahrscheinlichkeitsverteilung auf der Menge aller Syntaxbäume induzieren.*** Oder anders formuliert, die Menge der Syntaxbäume $\mathcal{T}(G)$ soll ein sicheres Ereignis sein:

$$p\left(\mathcal{T}(G)\right) = \sum_{x \in \mathcal{T}(G)} p(x) = 1 \ .$$

Nur eine einfache Folgerung aus der Definition der Satzwahrscheinlichkeit, als Ereigniswahrscheinlichkeit von mindestens einer ihrer Analysen, ist dann: ***Sind die Grammatikregel-Wahrscheinlichkeiten so gewählt, dass sie eine Wahrscheinlichkeitsverteilung auf der Menge aller Syntaxbäume induzieren, dann induzieren sie auch eine Wahrscheinlichkeitsverteilung auf der Sprache der Grammatik.*** Oder anders ausgedrückt, die Sprache $\mathcal{L}(G)$ ist dann ein sicheres Ereignis:

$$p\left(\mathcal{L}(G)\right) = \sum_{y \in \mathcal{L}(G)} p(y) = \sum_{y \in \mathcal{L}(G)} \sum_{x \in \mathcal{T}(y)} p(x) = \sum_{x \in \mathcal{T}(G)} p(x) = 1 \ .$$

In Booth und Thompson (1973) finden sich hinreichende Kriterien, denen die Grammatikregel-Wahrscheinlichkeiten genügen müssen, damit diese Anforderung erfüllt wird. Zum sogenanten *Standard-Wahrscheinlichkeitsmodell* für kontextfreie Grammatiken gehört die folgende Nebenbedingung: Für alle *produktiven* und *erreichbaren* Grammatikkategorien A gelte:

$$\sum_{\substack{r \in G, \\ \mathrm{lhs}(r) = A}} p(r) = 1 \ .$$

Diese Nebenbedingung ist auf den ersten Blick unintuitiv. Offensichtlich ist die folgende Relation für Grammtikregeln $r,\ r' \in G$:

$$r \sim r' \quad :\Leftrightarrow \quad \mathrm{lhs}(r) = \mathrm{lhs}(r')$$

reflexiv, transitiv und *symmetrisch*, also eine *Äquivalenzrelation*. Eine gegebene kontextfreie Grammatik wird damit in *Äquivalenzklassen*

$$G_A := \{r \in G \mid \mathrm{lhs}(r) = A\}$$

eingeteilt und die Standard-Nebenbedingung für probabilistische kontextfreie Grammatiken (ohne unproduktive und unerreichbare Grammatikkategorien) besagt, dass die Gram-

matikregel-Wahrscheinlichkeiten auf den Äquivalenzklassen G_A eine Wahrscheinlichkeitsverteilung bilden:

$$p(G_A) = \sum_{r \in G_A} p(r) = 1 \qquad \forall A \; .$$

Somit sollte also nicht von einer Grammatikregel-Wahrscheinlichkeit $p(r)$ gesprochen werden, sondern besser von den Grammatikregel-Wahrscheinlichkeitsverteilungen $p_A(r) = p_{G_A}(r)$ (wobei A eine Grammatikkategorie ist, die auf der linken Seite irgendeiner Grammatikregel auftreten kann).[8]

Die Beispielgrammatik aus Abbildung 2.4 verletzt die Standard-Nebenbedingung, weil sich ihre Grammatikregel-Wahrscheinlichkeiten auf den Äquivalenzklassen:

$$G_S = \{S \rightarrow NP \; schläft, \;\; S \rightarrow Hans\}$$

und:

$$G_{NP} = \{NP \rightarrow Hans\}$$

zu den Werten 1.7 und 0.3 aufsummieren. Die Spielgrammatik beweist also, dass die Standard-Nebenbedingung für probabilistische Grammatiken nicht notwendig ist, um eine Wahrscheinlichkeitsverteilung auf $\mathcal{T}(G)$ zu induzieren. Der folgende Satz von Booth und Thompson (1973) zeigt, dass die Standard-Nebenbedingung auch keine hinreichende Bedingung ist.[9]

Ferner entnimmt man dem Satz, dass eine relativ einfache zusätzliche Nebenbedingung garantiert, dass die probabilistische kontextfreie Grammatik *konsistent* ist, d.h. eine Wahrscheinlichkeitsverteilung auf der Sprache $\mathcal{L}(G)$ induziert.

Theorem(Booth und Thompson 1973): Sei G eine probabilistische kontextfreie Grammatik, welche die Standard-Nebenbedingung erfüllt. Für jede Grammatikkategorie A sei

[8] Es ist vielleicht hilfreich, darauf hinzuweisen, dass eine probabilistische kontextfreie Grammatik mit Standard-Nebenbedingung als stochastische Modellierung eines Generators für kontextfreie Grammatiken angesehen werden kann, der in jedem Schritt eine Regel für das nächste zu expandierende Nicht-Terminal auswählt.

[9] Ein Beispiel für eine solche probabilistische kontextfreie Grammatik ist:

$$\begin{array}{ll} S \rightarrow S \; S & (p) \\ S \rightarrow a & (1-p) \end{array}$$

Die Sprache dieser Grammatik besteht aus den Sätzen a, aa, aaa, $aaaa$, etc. Es ist relativ leicht zu zeigen, dass für genügend grosse Werte $0 < p < 1$ die Wahrscheinlichkeiten der Synatxbäume aller dieser Sätze sich nicht auf 1 aufsummieren können. Eine genaue Analyse zeigt, dass dies daran liegt, dass die erwartete Tiefe der Syntaxbäume unendlich gross wird. Der Satz von Booth und Thompson (1973) gibt für probabilistische kontextfreie Grammatiken mit Standard-Nebenbedingung eine zusätzliche Nebenbedingung an, die gerade dieses Verhalten verhindert, d.h. eine endliche erwartete Tiefe von Syntaxbäumen erzwingt.

eine *Grammatikregel-Generierungsfunktion*:

$$g_A : [0,1]^{|\mathcal{C}|} \to \mathbb{R}$$

durch:

$$g_A(x) := \sum_{r \in G_A} p(r) \cdot \prod_{B \in \mathcal{C}} x_B^{f_B(r)}$$

definiert. Hierbei ist $\mathcal{C}$ die geordnete Menge aller Grammatikkategorien und $f_B(r)$ die Vorkommenshäufigkeit der Grammatikkategorie B in der rechten Seite rhs(r) der Grammatikregel r. Seien $|\lambda_1| \geq \ldots \geq |\lambda_{|\mathcal{C}|}|$ die absteigend geordneten Beträge aller *Eigenwerte* der *Funktionalmatrix*:

$$\left(\frac{\partial\, g_A}{\partial\, x_B}\right)_{A,B \in \mathcal{C}}$$

an der Stelle $x =< 1 \ldots 1 >$. Dann gilt:

$$\begin{aligned} |\lambda_1| < 1 \quad &\Rightarrow \quad G \text{ ist konsistent ,} \\ |\lambda_1| > 1 \quad &\Rightarrow \quad G \text{ ist nicht konsistent .} \end{aligned}$$

Bemerkungen: Das Theorem macht für den Fall $|\lambda_1| = 1$ keine Aussage. Ein Wert $\lambda \in \mathbb{R}$ heisst *Eigenwert* einer Matrix $\mathcal{M}$, falls es einen (***spaltenweisen***) *Eigenvektor* $x \neq 0$ gibt, mit[10]:

$$\mathcal{M} \cdot x = \lambda \cdot x \ .$$

Da die Funktionalmatrix $\left(\frac{\partial\, g_A}{\partial\, x_B}\right)_{A,B \in \mathcal{C}}$ nicht symmetrisch ist, soll klargestellt werden, dass ihre ***Zeilen*** aus den sogenannten *Gradienten* der Grammatikregel-Generierungsfunktionen bestehen. Der Gradient ∇ ist der (***zeilenweise***) Vektor der *ersten partiellen Ableitungen* einer Funktion nach allen ihren Argumenten; hier:

$$\nabla\, g_A := \left\langle \frac{\partial\, g_A}{\partial\, x_B} \middle| B \in \mathcal{C} \right\rangle \ .$$

Es gilt:

$$\left(\frac{\partial\, g_A}{\partial\, x_B}\right)_{A,B \in \mathcal{C}} = \left(\nabla\, g_A \right)_{A \in \mathcal{C}} \ .$$

Laut Booth und Thompson (1973) repräsentiert die *erste partielle Ableitung*

$$\left.\frac{\partial\, g_A}{\partial\, x_B}\right|_{x=<1\ldots1>}$$

[10] Aus der linearen Algebra ist bekannt, dass eine Matrix höchstens $|\mathcal{C}|$ Eigenwerte haben kann. Die *Vielfachheit* eines Eigenwertes ist die Dimension des Vektorraums der Eigenvektoren dieses Eigenwertes. Alle Vielfachheiten addieren sich (für komplexe Vektorräume) zu der Zahl $|\mathcal{C}|$ auf.

$$
\begin{array}{lll}
S \to Y_1^c \ldots Y_d^c & (p_c) & \text{für alle } c \in C \\
Y_1^c \to y_1 & (p_{y_1}^{(c)}) & \text{für alle } c \in C,\ y_1 \in \mathcal{Y}_1 \\
\vdots & & \\
Y_d^c \to y_d & (p_{y_d}^{(c)}) & \text{für alle } c \in C,\ y_d \in \mathcal{Y}_d
\end{array}
$$

Abbildung 2.6: *Probabilistisches d-dimensionales Clustering mit $|C|$ Klassen* mit Hilfe einer probabilistischen kontextfreien Grammatik ($d \geq 2$).

die erwartete[11] Vorkommenshäufigkeit der Grammatikkategorie B in der Äquivalenzklasse G_A.

Chi beweist (Chi (1998), Chi (1999)), dass Baumbank-Grammatiken und die mit dem Inside-Outside-Algorithmus estimierten Grammatiken konsistent sind, wenn keine zusätzlichen Techniken, wie Smoothing von Grammatikregeln, etc. eingesetzt werden. Leider ist dies fast immer der Fall.

Dem Autor ist nicht bekannt, dass das Kriterium aus dem Theorem von Booth and Thompson jemals im Rahmen einer echten Anwendung angewandt worden wäre. Daher ist es möglich, dass einige der in der Praxis trainierten und angewandten kontextfreien Grammatiken nicht konsistent sind; Es könnte daher sehr vielversprechend sein, zu untersuchen, ob ein Zusammenhang zwischen Konsistenz und Güte kontextfreier Grammatiken besteht. Leider geht dies über den Rahmen dieser Arbeit hinaus.

Im folgenden wird die Konsistenz einer probabilistischen kontextfreien Grammatik bewiesen, welche in Kapitel 4 genauer vorgestellt werden wird. Dort wird gezeigt, dass mit Hilfe der Konsistenz auf rein maschinellem Weg möglich sein wird, einen Korpus von d-dimensionalen linguistischen Daten probabilistisch zu klassifizieren. Die Abbildung 2.6 zeigt die Grammatik G_{cluster} mit $|C|+|C|\cdot|\mathcal{Y}_1|+\ldots+|C|\cdot|\mathcal{Y}_d|$ Grammatikregeln, bestehend aus den $1+|C|\cdot d$ Grammatikkategorien:

$$S,\ Y_i^c,\ (c \in C,\ i = 1 \ldots d),$$

und dem Grammatiklexikon:

$$\mathcal{Y}_1 + \ldots + \mathcal{Y}_d\ .$$

Das $'+'$-Zeichen steht für die Vereinigung von disjunkten Mengen; es wird also vorausgesetzt, dass jedes Wort (jedes Fragment) eines linguistischen Datums nur in einer ganz bestimmten Koordinate vorkommen kann. Dies ist keine Beschränkung an Allgemeinheit, da durch eine Kodierung, welche die Koordinaten mitberücksichtigt, stets erreicht werden kann, dass die Daten die vorausgesetzte Eigenschaft tragen.

[11] Erwartungswerte werden in Abschnitt 2.3 definiert.

Die Sprache der Grammatik ist:

$$\mathcal{L}(G_{\text{cluster}}) = \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_d = \prod_{i=1}^{d} \mathcal{Y}_i \ .$$

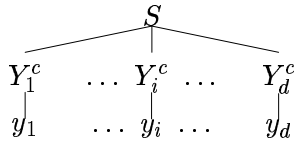
In der Grammatik G_{cluster} hat jeder Satz $y =< y_1 \ldots y_d >$ genau $|C|$ nicht-rekursive Syntaxbäume:

$$\mathcal{T}(y) = \left\{ \quad \begin{array}{ccccc} & & S & & \\ & \swarrow & \downarrow & \searrow & \\ Y_1^c & \ldots & Y_i^c & \ldots & Y_d^c \\ \downarrow & & \downarrow & & \downarrow \\ y_1 & \ldots & y_i & \ldots & y_d \end{array} \quad \middle| \quad c \in C \right\}$$

Die stochastischen Aspekte dieser Grammatik sind: Standard-Nebenbedingung, Syntaxbaum- und Satzwahrscheinlichkeiten. Die Standard-Nebenbedingung hat hier die Gestalt:

$$\sum_{c \in C} p_c = 1, \qquad \sum_{y_i \in \mathcal{Y}_i} p_{y_i}^{(c)} = 1, \ (c \in C, \ i = 1 \ldots d) \ .$$

Ein Syntaxbaum:



hat die Wahrscheinlichkeit:

$$p_c \cdot p_{y_1}^{(c)} \cdots p_{y_d}^{(c)} \ .$$

Die Wahrscheinlichkeit eines Satzes $y =< y_1 \ldots y_d >$ ist daher:

$$\sum_{c \in C} p_c \cdot p_{y_1}^{(c)} \cdot \ldots \cdot p_{y_d}^{(c)} \ .$$

Das Theorem (Booth und Thompson 1973) verlangt die Berechnung der Grammatikregel-Generierungsfunktionen und ihrer Gradienten. Es gilt gemäss Definition:

$$\begin{aligned} g_S(x) &= \sum_{r \in G_S} p(r) \cdot \prod_B x_B^{f_B(r)} \\ &= \sum_{c \in C} p(S \to Y_1^c \ldots Y_d^c) \cdot \prod_B x_B^{f_B(S \to Y_1^c \ldots Y_d^c)} \\ &= \sum_{c \in C} p_c \cdot x_{Y_1^c} \cdots x_{Y_d^c} \ , \end{aligned}$$

denn $f_B(S \to Y_1^c \ldots Y_d^c) = 1$ für $B = Y_i^c$. Für den Gradienten gilt daher:

$$\frac{\partial \ g_S}{\partial \ x_B} = \left\{ \begin{array}{ll} 0 & \text{falls } B = S \\ p_c \cdot x_{Y_1^c} \cdots x_{Y_{i-1}^c} \cdot x_{Y_{i+1}^c} \cdots x_{Y_d^c} & \text{falls } B = Y_i^c \end{array} \right.$$

An der Stelle $x =< 1 \ldots 1 >$ gilt daher:

$$\frac{\partial \ g_S}{\partial \ x_B} = \left\{ \begin{array}{ll} 0 & \text{falls } B = S \\ p_c & \text{falls } B = Y_i^c \end{array} \right.$$

Mit der Ordnung

$$\mathcal{C} = \left\langle S\ Y_1^1\ \ldots Y_d^1\ \ldots .\ Y_1^{|C|}\ \ldots Y_d^{|C|} \right\rangle$$

gilt für das Startsymbol S, dass:

$$\nabla\ g_S = \left\langle 0\ p_1 \ldots p_1\ \ldots .\ p_{|C|} \ldots p_{|C|} \right\rangle$$

Ferner gilt per Definition:

$$\begin{aligned} g_{Y_i^c}(x) &= \sum_{r \in G_{Y_i^c}} p(r) \cdot \prod_B x_B^{f_B(r)} \\ &= \sum_{y_i \in \mathcal{Y}_i} p(Y_i^c \to y_i) \cdot \prod_B x_B^{f_B(Y_i^c \to y_i)} \\ &= \sum_{y_i \in \mathcal{Y}_i} p_{y_i}^{(c)} \cdot 1 \\ &= 1\ , \end{aligned}$$

sodass für jede Grammatikkategorie Y_i^c gilt, dass:

$$\nabla\ g_{Y_i^c} = 0\ .$$

Die Funktionalmatrix hat somit die folgende Gestalt:

$$\begin{pmatrix} 0\ p_1 \ldots p_1 & \ldots . & p_{|C|} \ldots p_{|C|} \\ 0 & \ldots . & 0 \\ \vdots & & \\ 0 & \ldots . & 0 \end{pmatrix}\ .$$

Zur Bestimmung von Eigenwerten $\lambda \in \mathbb{R}$ ist also das folgende Gleichungssystem zu lösen:

$$\begin{pmatrix} 0\ p_1 \ldots p_1 & \ldots . & p_{|C|} \ldots p_{|C|} \\ 0 & \ldots . & 0 \\ \vdots & & \\ 0 & \ldots . & 0 \end{pmatrix} \cdot \begin{pmatrix} x_S \\ x_{Y_1^1} \\ \ldots \\ x_{Y_d^1} \\ \vdots \\ x_{Y_1^{|C|}} \\ \ldots \\ x_{Y_d^{|C|}} \end{pmatrix} = \lambda \cdot \begin{pmatrix} x_S \\ x_{Y_1^1} \\ \ldots \\ x_{Y_d^1} \\ \vdots \\ x_{Y_1^{|C|}} \\ \ldots \\ x_{Y_d^{|C|}} \end{pmatrix}\ .$$

Dies ist äquivalent zu:

$$\begin{pmatrix} \sum_{c\in C} p_c \cdot \sum_{i=1}^{d} x_{Y_i^c} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \lambda \cdot \begin{pmatrix} x_S \\ x_{Y_1^1} \\ \cdots \\ x_{Y_d^1} \\ \vdots \\ x_{Y_1^{|C|}} \\ \cdots \\ x_{Y_d^{|C|}} \end{pmatrix} .$$

Für ein $\lambda \neq 0$ folgt zunächst:

$$x_{Y_i^c} = 0, \quad (c \in C,\ i = 1 \ldots d) ,$$

und anschliessend:

$$x_S = 0 ,$$

sodass nur $\lambda = 0$ ein Eigenwert der Funktionalmatrix der Grammatikregel-Generierungsfunktionen ist. Nach dem Satz von Booth und Thompson (1973) ist die Grammatik aus Abbildung 2.6, unabhängig von der speziellen Wahl der Grammatikregel-Wahrscheinlichkeiten, eine konsistente Grammatik. Wegen der Einfachheit dieser Grammatik, die ja keine rekursiven Strukturen aufweist, kann dieses Ergebnis auch auf direktem Weg verifiziert werden. Hierzu wird auf das Kapitel 4 verwiesen. Insbesondere für rekursive kontextfreie Grammatiken ist der Satz von Booth und Thompson (1973) jedoch das einzige zur Verfügung stehende Werkzeug, um Konsistenz zu beweisen.

Zum Abschluss dieses Abschnitts soll noch der für den EM-Algorithmus wichtige Begriff der bedingten Wahrscheinlichkeit eingeführt werden. Ist X kein *unmögliches Ereignis*, d.h. gilt nicht $p(X) = 0$, so kann eine sogenannte *bedingte Wahrscheinlichkeitsverteilung* $p(.|X)$ auf $\mathcal{X}$:

$$p(.|X) : \mathcal{X} \to [0, 1]$$

eingeführt werden, indem die *bedingte Wahrscheinlichkeit eines Datentyps* $x \in \mathcal{X}$ wie folgt definiert wird[12]:

$$p(x|X) = \begin{cases} p(X)^{-1} \cdot p(x) & \text{falls } x \in X \\ 0 & \text{sonst} \end{cases}$$

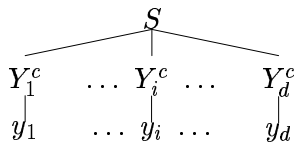
[12]Die bedingte Wahrscheinlichkeitsverteilung kann auf $\mathcal{P}(\mathcal{X})$ fortgesetzt werden. Das Ergebnis ist:

$$p(E|X) = p(X)^{-1} \cdot p(E \cap X) \text{ für alle } E \in \mathcal{P}(\mathcal{X}) ,$$

und findet sich in allen Lehrbüchern zur Stochastik.

Man prüft sehr einfach nach, dass $\sum_{x \in \mathcal{X}} p(x|X) = 1$ gilt, d.h. dass tatsächlich eine Wahrscheinlichkeitsverteilung auf $\mathcal{X}$ vorliegt. Offensichtlich nutzt man hierzu aus, dass $\sum_{x \in X} p(x|X) = 1$ ist, d.h. dass $p(.|X)$ schon eine Wahrscheinlichkeitsverteilung auf dem linguistischen Ereignis $X \subseteq \mathcal{X}$ selbst ist.

Der, zu der Grammatik aus Abbildung 2.6 gehörende Syntaxbaums x_c, $(c \in C)$:



hat gegeben seinem Satz $y_1 \dots y_d$ beispielsweise die folgende bedingte Wahrscheinlichkeit:

$$\begin{aligned} p(x_c|y_1 \dots y_d) &= p(x_c|\{x_1 \dots x_{|C|}\}) \\ &= \frac{p(x_c)}{p(\{x_1 \dots x_{|C|}\})} \\ &= \frac{p(x_c)}{\sum_{c \in C} p(x_c)} \\ &= \frac{p_c \cdot p_{y_1}^{(c)} \cdots p_{y_d}^{(c)}}{\sum_{c \in C} p_c \cdot p_{y_1}^{(c)} \cdots p_{y_d}^{(c)}} \; . \end{aligned}$$

Mit diesem Beispiel soll dieser Abschnitt beschlossen werden.

2.3 Erwartungswert und Korpusperplexität

Eine Funktion $f : \mathcal{Y} \to \mathbb{R}$ wird *verallgemeinerte Häufigkeitsverteilung* genannt. Zu der Menge der verallgemeinerten Häufigkeitsverteilungen:

$$\mathcal{F}(\mathcal{Y}) = \{f : \mathcal{Y} \to \mathbb{R}\}$$

gehören insbesondere die natürlichen Häufigkeitsverteilungen. Im Gegensatz zu natürlichen Häufigkeitsverteilungen kann eine verallgemeinerte Häufigkeitsverteilung zum einen für unendlich viele $y \in \mathcal{Y}$ einen Wert $f(y) \neq 0$, und zum anderen sogar negative oder nichtganzzahlige Werte annehmen.

Sei $p(.)$ eine Wahrscheinlichkeitsverteilung. Der Wert:

$$p[f] = \sum_{y \in \mathcal{Y}} p(y) \cdot f(y)$$

wird *Erwartungswert von f bezüglich p* genannt.

Aus mathematischer Sicht stecken in dieser Definition zwei Schwierigkeiten. Erstens kann die Reihe $\sum_{y \in \mathcal{Y}} p(y) \cdot f(y)$ aus abzählbar unendlich vielen Termen bestehen, sodass man fordern sollte, dass die Reihe konvergiert. Zweitens muss das Summationsergebnis von der Reihenfolge der Summation unabhängig sein. Beide Schwierigkeiten führen dazu, dass definiert wird:

$$\begin{aligned} p[f] \text{ existiert} \quad &:\Leftrightarrow \quad \sum_{y \in \mathcal{Y}} p(y) \cdot f(y) \text{ ist absolut konvergent} \\ &\Leftrightarrow \quad \sum_{y \in \mathcal{Y}} p(y) \cdot |f(y)| < \infty \ . \end{aligned}$$

Die wichtigste Eigenschaft einer *Erwartung* $p[.]$ ist deren *Linearität*: für je zwei reelle Zahlen $\alpha, \beta \in \mathbb{R}$ und für je zwei verallgemeinerte Häufigkeitsverteilungen $f, g \in \mathcal{F}(\mathcal{Y})$, deren Erwartungswerte $p[f]$ und $p[g]$ existieren, existiert auch der Erwartungswert der verallgemeinerten Häufigkeitsverteilung:

$$\alpha f + \beta g : \mathcal{Y} \to \mathbb{R}, \quad y \mapsto \alpha \cdot f(y) + \beta \cdot g(y)$$

und es gilt:

$$p\,[\ \alpha f + \beta g\ ] = \alpha \cdot p\,[\ f\ ] + \beta \cdot p\,[\ g\ ] \ .$$

Diese Eigenschaft ist einfach zu beweisen und wird oft verwendet werden.

Leider wird in den Lehrbüchern zur Stochastik nie die schöne *zweite Linearitätseigenschaft* der Erwartung erwähnt: für je zwei reelle Zahlen $\alpha, \beta \geq 0$ mit $\alpha + \beta = 1$ und für je zwei Wahrscheinlichkeitsverteilungen $p(.)$ und $q(.)$, sowie für eine beliebige verallgemeinerte

Häufigkeitsverteilung $f \in \mathcal{F}(\mathcal{Y})$, deren Erwartungswerte $p[f]$ und $q[f]$ existieren, existiert auch der Erwartungswert $(\alpha p + \beta q)[f]$ und es gilt:

$$(\alpha p + \beta q)[f] = \alpha \cdot p[f] + \beta \cdot q[f] \ .$$

Natürlich ist auch die zweite Linearitätseigenschaft einfach zu beweisen, wenn man nur beachtet, dass

$$\alpha p + \beta q : \ \mathcal{Y} \to [0,1], \quad y \mapsto \alpha \cdot p(y) + \beta \cdot q(y)$$

wegen $\alpha + \beta = 1$, eine Wahrscheinlichkeitsverteilung ist.

Die kleine Auflistung formaler Eigenschaften von Erwartungswerten soll mit der Bemerkung abgeschlossen werden, dass sich Erwartungswerte merkwürdig symmetrisch verhalten, wenn die verallgemeinerte Häufigkeitsverteilung eine Wahrscheinlichkeitsverteilung ist:

$$p\,[\,q\,] = q\,[\,p\,] \ .$$

Der Beweis ist trivial.

Erwartungswerte sind wichtige Grössen in der Stochastik, wie man sich leicht überzeugen kann, indem man den zentralen Begriff des ersten Abschnitts, d.h. den Begriff der Korpuswahrscheinlichkeit, etwas genauer untersucht:

$$p(f) = \prod_{y \in \mathcal{Y}} p(y)^{f(y)} \ .$$

Nach Logarithmierung (mit einer beliebigen logarithmischen Basis log-base $\in \{2, e, 10, \ldots\}$) wird aus dem Produkt eine Summe:

$$\begin{aligned} \log\ p(f) &= \sum_{y \in \mathcal{Y}} \log\ p(y)^{f(y)} \\ &= \sum_{y \in \mathcal{Y}} f(y) \cdot \log\ p(y) \ . \end{aligned}$$

Offensichtlich ist die Umformung formal richtig, doch tritt an dieser Stelle ein kleines Problem auf: wenn die Wahrscheinlichkeit $p(y)$ gleich 0 ist, ist der Logarithmus $\log\ p(y)$ gleich $-\infty$. Ist gleichzeitig die Häufigkeit $f(y)$ gleich 0, so ist zu klären, welchen Wert das Produkt $f(y) \cdot \log\ p(y)$, also $0 \cdot (-\infty)$ annehmen soll. Ein Blick auf die Korpuswahrscheinlichkeit $p(f)$ zeigt, dass das problematische y mit dem Wahrscheinlichkeitsfaktor $p(y)^{f(y)} = 0^0 = 1$ in die Korpuswahrscheinlichkeit eingeht. Derartige Datentypen werden also bei der Berechnung der Korpuswahrscheinlichkeit ganz einfach ignoriert. Dies spricht natürlich dafür, dass

$$0 \cdot (-\infty) := 0$$

definiert wird, sodass problematische Datentypen auch bei der Berechnung der logarithmierten Korpuswahrscheinlichkeit ignoriert werden. Eine mathematische Rechtfertigung dieser Definition ergibt sich aus dem Fakt, dass die Funktion $x \cdot \log\ x$ für $x \to 0$ gegen 0 konvergiert.

Nach Division mit der absoluten Häufigkeit erhält man die sogenannte *Average-Log-Likelihood*, die auch kurz *Log-Likelihood* genannt wird:

$$L(p;\tilde{p}) := |f|^{-1} \cdot \log\ p(f)\ .$$

Es folgt:

$$\begin{aligned} L(p;\tilde{p}) &= |f|^{-1} \cdot \sum_{y \in \mathcal{Y}} f(y) \cdot \log\ p(y) \\ &= \sum_{y \in \mathcal{Y}} |f|^{-1} \cdot f(y) \cdot \log\ p(y) \\ &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \cdot \log\ p(y) \\ &= \tilde{p}[\log\ p]\ . \end{aligned}$$

Wegen der durchgeführten Division mit der absoluten Häufigkeit $|f|$ ist die Log-Likelihood für zwei Korpora identisch, wenn diese dieselbe empirische Wahrscheinlichkeitsverteilung besitzen. Diese Eigenschaft ist an sich relativ bedeutungslos, hat aber doch für Korpora Konsequenzen, in denen dieselben Datentypen mit denselben empirischen Wahrscheinlichkeiten, jedoch mit unterschiedlichen Häufigkeiten vorkommen: Kommt zum Beispiel jedes Datentoken aus einem Korpus exakt x-mal, z.B. tausend- oder zehntausend-mal, in einem zweiten Korpus vor, so unterscheiden sich zwar die Korpuswahrscheinlichkeiten beider Korpora, doch die Log-Likelihood ist für beide Korpora identisch. Die Log-Likelihood sollte daher immer entweder auf der Grundlage des kleineren Korpus $f(.)$ oder der empirischen Wahrscheinlichkeitsverteilung $\tilde{p}(.)$ berechnet werden. Die Korpuswahrscheinlichkeit hingegen sollte auf der Basis der Log-Likelihood (oder einer äquivalenten Grösse), sowie der absoluten Häufigkeit berechnet werden.

Wie man sich leicht überlegt, ist der negative Erwartungswert $-\tilde{p}[\log\ p]$ eine nicht-negative reelle Zahl aus dem Intervall $[0, \infty]$. Sie wird *Cross-Entropie* genannt:

$$\begin{aligned} H_{\text{cross}}(p;\tilde{p}) &:= -\sum_{y \in \mathcal{Y}} \tilde{p}(y) \cdot \log\ p(y) \\ &= -\tilde{p}[\log\ p]\ . \end{aligned}$$

Es gilt natürlich einfach:

$$H_{\text{cross}}(p;\tilde{p}) = -L(p;\tilde{p})\ ,$$

sodass Cross-Entropie und Average-Log-Likelihood alternativ zu gebrauchende Begriffe sind, je nachdem, ob man negative oder positive Zahlen bevorzugt. Nach Umkehrung von Division und Logarithmierung erhält man aus der Log-Likelihood wieder die Korpuswahrscheinlichkeit. Es folgt:

$$\begin{aligned} p(f) &= \text{log-base}^{\,|f|\cdot L(p;\tilde{p})} \\ &= \text{log-base}^{\,|f|\cdot \tilde{p}[\log\ p]} \\ &= \text{log-base}^{\,-\,|f|\cdot H_{\text{cross}}(p;\tilde{p})}\ . \end{aligned}$$

Rein formal gesehen, lässt sich die Korpuswahrscheinlichkeit also durch einen Erwartungswert bezüglich der empirischen Wahrscheinlichkeitsverteilung berechnen, was die Bedeutung des Begriffs des Erwartungswertes an sich, der Log-Likelihood oder der Cross-Entropie im besonderen, sowie der empirischen Wahrscheinlichkeitsverteilung in das rechte Licht rückt.

Besonders erfreulich an dieser Formel ist, dass die Cross-Entropie und die Log-Likelihood sofern sie ihrer Natur nach überhaupt von einem Korpus unabhängig sein können, lediglich von der empirischen Wahrscheinlichkeiten der Datentypen im Korpus, nicht aber von deren absoluten Häufigkeiten oder der absoluten Häufigkeit aller Datentypen im Korpus abhängen. Deshalb ist es besonders störend, dass die Cross-Entropie und die Log-Likelihood von einem recht arbiträren Wert, wie der logarithmischen Basis log-base , abhängig sind. Durch die Einführung der sogenannten *Korpusperplexität* kann erreicht werden, dass die Korpuswahrscheinlichkeit auf eine noch natürlichere Grösse zurückgeführt wird:

$$\text{perplexity}(p;\tilde{p}) := \text{log-base}^{\,H_{\text{cross}}(p;\tilde{p})}\ .$$

In der Regel wird die Korpusperplexität für die logarithmische Basis log-base $= 2$ definiert; man kann sich allerdings leicht davon überzeugen, dass die Korpusperplexität von der speziellen Wahl der logarithmischen Basis log-base unabhängig ist: etwas später wird gezeigt werden, dass sich die Korpusperplexität auf die zwei einzelnen Werte der Korpuswahrscheinlichkeit und der Korpuslänge zurückführen lässt. Da diese beiden Werte von der speziellen Wahl der logarithmischen Basis unabhängig sind, gilt dies auch für die Korpusperplexität.

Andererseits ist die Korpusperplexität allein durch die logarithmische Basis log-base und die Cross-Entropie definiert. Somit ist die Korpusperplexität auch von der Korpuslänge unabhängig.

Beide Eigenschaften zusammen (Unabhängigkeit von der Korpuslänge und Unabhängigkeit von der logarithmischer Basis) machen die Korpusperplexität zu einer besseren Messgrösse für die Korpuswahrscheinlichkeit als die Log-Likelihood oder die Cross-Entropie, die zwar beide von der Korpuslänge nicht aber von der logarithmischen Basis unabhängig sind.

Paradoxerweise ist die Korpusperplexität in einem gewissen Sinn sogar ein besserer Wert als die Korpuswahrscheinlichkeit selbst: Da die Korpuswahrscheinlichkeit von der Korpuslänge abhängig ist, ist sie im Gegensatz zur Korpusperplexität leider nicht zum Vergleich von Korpora unterschiedlicher Länge geeignet (dies gilt sogar auch, wenn diese dieselbe empirische Wahrscheinlichkeitsverteilung aufweisen).

Die Korpusperplexität ist eine Funktion der Wahrscheinlichkeitsverteilung $p(.)$ der Datentypen im Korpus, sowie des Korpus $\tilde{p}(.)$. Daher kann die Korpusperplexität auf zwei Weisen interpretiert werden:

Die plausiblere Interpretation der Korpusperplexität ist: die durchschnittliche Anzahl (oder besser die erwartete Anzahl) von Datentoken, die an einer festen (aber beliebigen) Position im Korpus vorkommen werden. Hierbei wird die Wahrscheinlichkeitsverteilung $p(.)$ in schon gewohnter Weise als Generator von Datentypen angesehen. Beispielsweise bedeutet eine Korpusperplexität von 39, dass der Generator durchschnittlich 39 verschiedene Datentoken an den Positionen im Korpus erzeugen wird. Oder anders formuliert: Jedes Datentoken wurde an seiner Position im Korpus von dem Generator mit einer durchschnittlichen Wahrscheinlichkeit $\frac{1}{39}$ erzeugt. Je kleiner die Korpusperplexität ist, desto besser ist ein Generator an den vorgegebenen Korpus angepasst.

Um diese Interpretation zu verifizieren, muss lediglich der Zusammenhang zwischen Korpuswahrscheinlichkeit und Korpusperplexität untersucht werden. Offenbar gilt:

$$\begin{aligned} p(f) &= \text{log-base}^{\,-|f|\cdot H_{\text{cross}}(p;\tilde{p})} \\ &= \left(\text{log-base}^{\,H_{\text{cross}}(p;\tilde{p})}\right)^{-|f|} \\ &= \text{perplexity}(p;\tilde{p})^{-|f|} \\ &= \left(\frac{1}{\text{perplexity}(p;\tilde{p})}\right)^{|f|} . \end{aligned}$$

Wenn vorausgesetzt wird, dass jedes Datentoken im Korpus mit der Gleichverteilung

$$\pi : \mathcal{Y} \to [0,1], \quad y \mapsto |\mathcal{Y}|^{-1}$$

erzeugt wird, kann die Korpuswahrscheinlichkeit explizit berechnet werden:

$$\begin{aligned} \pi(f) &= \prod_{y\in\mathcal{Y}} \pi(y)^{f(y)} \\ &= \prod_{y\in\mathcal{Y}} \pi^{f(y)} \\ &= \pi^{\sum_{y\in\mathcal{Y}} f(y)} \\ &= \pi^{|f|} \\ &= \left(\frac{1}{|\mathcal{Y}|}\right)^{|f|} . \end{aligned}$$

Ein Vergleich der letzten beiden Resultate zeigt, dass die Korpusperplexität in dieser Situation gleich der Anzahl $|\mathcal{Y}|$ aller Datentypen ist:

$$\text{perplexity}(\pi) = |\mathcal{Y}| \ .$$

Natürlich kann dasselbe Ergebnis mit den Definitionen von Cross-Entropie und Korpusperplexität erzielt werden:

$$\begin{aligned} H_{\text{cross}}(\pi) &:= -\tilde{p}[\log \pi] \\ &= -\log \pi \quad (\text{denn } \pi \text{ ist konstant}), \end{aligned}$$

$$\begin{aligned} \text{perplexity}(\pi) &:= \text{log-base}^{\,H_{\text{cross}}(\pi)} \\ &= \text{log-base}^{\,-\log \pi} \\ &= \frac{1}{\pi} \\ &= |\mathcal{Y}| \ . \end{aligned}$$

Dieses Ergebnis korreliert sehr schön mit der oben angegebenen Interpretation der Korpusperplexität.

Im allgemeinen Fall gilt offenbar:

$$\text{perplexity}(p;\tilde{p}) = \sqrt[|f|]{\frac{1}{p(f)}} \ .$$

Die zweite Interpretation der Korpusperplexität betrifft den Fall, für den die Wahrscheinlichkeitsverteilung $p(.)$ als gegeben angesehen wird. Für eine fixierte Wahrscheinlichkeitsverteilung $p(.)$ schätzt die Korpusperplexität die Entropie eines beliebigen Korpus $\tilde{p}(.)$, welche ihrerseits als die Komplexität der im Korpus manifestierten Sprache interpretiert werden kann. Je grösser die Korpusperplexität ist, desto grösser ist die Korpus-Entropie.

Die *Korpus-Entropie* $H(\tilde{p})$ und die *relative Entropie* $D(\tilde{p}\|p)$, besser als *Kullback-Leibler-Distanz* bekannt, sind zwei weitere, sehr bekannte Erwartungswerte. Letztere ist definiert als:

$$D(\tilde{p}\|p) = \sum_{y\in\mathcal{Y}} \tilde{p}(y) \cdot \log \ \frac{\tilde{p}(y)}{p(y)} = \tilde{p}\left[\ \log \ \frac{\tilde{p}}{p} \ \right] \ ,$$

erstere als:

$$H(\tilde{p}) = -\sum_{y\in\mathcal{Y}} \tilde{p}(y) \cdot \log \ \tilde{p}(y) = -\tilde{p}\,[\ \log \ \tilde{p}\ ] \ .$$

Für problematische $y \in \mathcal{Y}$ werden die folgenden Konventionen benutzt, die sich aus Plausibilitäts- und Stetigkeits-Argumenten ergeben:

$$0 \cdot \log \frac{0}{p(y)} := 0, \qquad \text{und} \qquad \tilde{p}(y) \cdot \log \frac{\tilde{p}(y)}{0} := \infty \ .$$

Es ist leicht einzusehen, dass die Korpus-Entropie wie die Cross-Entropie eine nicht-negative reelle Zahl ist.

Die sogenannte *Informations-Ungleichung* (siehe z.B. Cover und Thomas (1991)) ist von fundamentaler Bedeutung für die Informationstheorie und zu ihrem Beweis sind tiefe Ergebnisse nötig. Sie besagt, dass auch die Kullback-Leibler-Distanz nicht-negativ ist (und dann genau Null, wenn $\tilde{p} = p$):

$$D(\tilde{p}\|p) \geq 0 \ .$$

In Abschnitt 2.4 wird die Informationsungleichung im Rahmen der Fragestellung, welche Wahrscheinlichkeitsverteilung einen gegebenen Korpus am besten repräsentiert, bewiesen werden. Die Antwort wurde schon in Abschnitt 2.1 nahegelegt: es ist natürlich die empirische Wahrscheinlichkeitsverteilung. Die folgenden Ergebnisse sind auch in diesem Zusammenhang sehr interessant.

Unter Benutzung der bekannten Rechengesetze für den Logarithmus und der Linearität des Erwartungswertes rechnet man leicht nach, dass:

$$\begin{aligned} D(\tilde{p}\|p) &= \tilde{p}\left[\ \log\ \frac{\tilde{p}}{p}\ \right] \\ &= \tilde{p}\,[\ \log\ \tilde{p} - \log\ p\ ] \\ &= \tilde{p}\,[\ \log\ \tilde{p}\ ] - \tilde{p}\,[\ \log\ p\ ] \\ &= -H(\tilde{p}) + H_{\text{cross}}(p;\tilde{p}) \ . \end{aligned}$$

Damit ist die folgende wichtige Aussage über das Verhältnis der drei Entropie-Grössen bewiesen: Die Cross-Entropie ist die Summe aus Korpus-Entropie und relativer Entropie:

$$H_{\text{cross}}(p;\tilde{p}) = H(\tilde{p}) + D(\tilde{p}\|p) \ .$$

Eine erste Konsequenz aus der Positivität der Kullback-Leibler-Distanz ist daher, dass die Cross-Entropie stets grösser ist als die Entropie:

$$H_{\text{cross}}(p;\tilde{p}) \geq H(\tilde{p}) \ .$$

Wenn man, analog zur Definition der Korpusperplexität zwei neue Grössen einführt:

$$\text{perplexity}_H(\tilde{p}) := \text{log-base}^{\ H(\tilde{p})} \ ,$$

sowie:

$$\text{perplexity}_D(p;\tilde{p}) := \text{log-base}^{\ D(\tilde{p}\|p)} \ ,$$

die man vielleicht *innere Korpusperplexität* und *relative Korpusperplexität* nennen könnte, so errechnet sich die Korpusperplexität elegant als das Produkt dieser zwei neuen Grössen:

$$\text{perplexity}(p;\tilde{p}) \quad = \quad \text{log-base}^{\ H(\tilde{p})+D(\tilde{p}\|p)}$$

$$
\begin{aligned}
&= \text{log-base}^{\,H(\tilde{p})} \cdot \text{log-base}^{\,D(\tilde{p}\|p)} \\
&= \text{perplexity}_H(\tilde{p}) \cdot \text{perplexity}_D(p;\tilde{p}) \ .
\end{aligned}
$$

Interessant ist, dass beide neu eingeführten Grössen, ebenso wie die Korpusperplexität , von der logarithmischen Basis (und natürlich von der Korpuslänge) unabhängig sind. Diese Eigenschaft haben ihre Basisgrössen Entropie und relative Entropie leider nicht. Wichtiger jedoch ist, dass sich die gute Interpretierbarkeit der Korpusperplexität , sowohl auf die innere Korpusperplexität als auch auf die relative Korpusperplexität überträgt. Hierzu dient:

$$
\begin{aligned}
p(f) &= \left(\frac{1}{\text{perplexity}(p;\tilde{p})}\right)^{|f|} \\
&= \left(\frac{1}{\text{perplexity}_H(\tilde{p}) \cdot \text{perplexity}_D(p;\tilde{p})}\right)^{|f|} \\
&= \left(\frac{1}{\text{perplexity}_H(\tilde{p})} \cdot \frac{1}{\text{perplexity}_D(p;\tilde{p})}\right)^{|f|} \ .
\end{aligned}
$$

Wenn die inverse Korpusperplexität, d.h. der Wert $\frac{1}{\text{perplexity}(p;\tilde{p})}$, als durchschnittliche Wahrscheinlichkeit interpretiert werden kann, mit der ein Prozess an einer festen (aber beliebigen) Position im Korpus ein Datentoken erzeugt, so kann, mit der eben hergeleiteten Formel für die Korpuswahrscheinlichkeit, dieser Prozess dahingehend interpretiert werden, dass er aus zwei stochastisch unabhängigen Modulen besteht:

Das erste Modul beschreibt nur den Korpus (oder wenn man so will, die Sprache, die in diesem speziellen Korpus codiert ist). Hierbei wird allein die innere Beschaffenheit des Korpus analysiert, ohne irgendwelche weitere, d.h. äussere Einflüsse zu berücksichtigen: mit der durchschnittlichen Wahrscheinlichkeit $\frac{1}{\text{perplexity}_H(\tilde{p})}$ liegt an einer bestimmten (aber beliebigen) Korpusposition ein bestimmtes Datentoken vor, wenn zur Bestimmung dieser Wahrscheinlichkeit nur der Korpus selbst, d.h. der Generator $\tilde{p}(.)$ benutzt wird. Wenn die Wahrscheinlichkeit $\frac{1}{\text{perplexity}_H(\tilde{p})}$ gross ist, dann ist der Korpus einfach, weil er "vorherberechenbar" ist. Ist die Wahrscheinlichkeit klein, dann ist der Korpus komplex, d.h. er ist "unvorherberechenbar".

Das zweite Modul beschreibt den Einfluss des Generators $p(.)$ auf den Generierungsprozess, wobei allerdings der Einfluss des ersten Moduls, d.h. die innere Beschaffenheit des Korpus , nicht mit berücksichtigt wird: ein bestimmtes Datentoken liegt dann an einer bestimmten (aber beliebigen) Korpusposition mit der durchschnittlichen Wahrscheinlichkeit $\frac{1}{\text{perplexity}_D(p;\tilde{p})}$ vor, wenn zur Bestimmung dieser Wahrscheinlichkeit zwar der Generator $p(.)$ benutzt wurde, jedoch zuvor die innere Beschaffenheit $\tilde{p}(.)$ des Korpus aus dem Generierungsprozess heraus dividiert wurde. Ist die Wahrscheinlichkeit $\frac{1}{\text{perplexity}_D(p;\tilde{p})}$ gross, dann ist der Korpus (modulo seiner inneren Beschaffenheit) einfach. Wenn die Wahrscheinlichkeit klein ist, dann ist der Korpus (in jedem Fall) komplex und unvorherberechenbar.

Ein Prozess, der stochastisch unabhängig aus diesen Einzelmodulen zusammengesetzt wird, erzeugt dann mit Wahrscheinlichkeit

$$\frac{1}{\text{perplexity}(p;\tilde{p})} = \frac{1}{\text{perplexity}_H(\tilde{p})} \cdot \frac{1}{\text{perplexity}_D(p;\tilde{p})}$$

an einer bestimmten (aber beliebigen) Korpusposition ein bestimmtes Datentoken.

Die hier angegebene Interpretation korreliert also sehr schön mit der oben angegebenen Formel zur Korpusperplexität.

Am Ende dieses Abschnitts sollen noch einmal alle bisher gefundenen Formeln für die Korpuswahrscheinlichkeit zusammengefasst werden:

$$\begin{aligned}
p(f) &= \prod_{y\in\mathcal{Y}} p(y)^{f(y)} \\
&= \text{log-base}^{\,|f|\cdot\tilde{p}[\log\ p]} \\
&= \text{log-base}^{\,|f|\cdot L(p;\tilde{p})} \\
&= \text{log-base}^{\,-\,|f|\cdot H_{\text{cross}}(p;\tilde{p})} \\
&= \text{log-base}^{\,-|f|\cdot(H(\tilde{p})+D(\tilde{p}\|p))} \\
&= \left(\frac{1}{\text{perplexity}(p;\tilde{p})}\right)^{|f|} \\
&= \left(\frac{1}{\text{perplexity}_H(\tilde{p})} \cdot \frac{1}{\text{perplexity}_D(p;\tilde{p})}\right)^{|f|} .
\end{aligned}$$

Für einen gegebenen Korpus $\tilde{p}(.)$ gelten daher die folgenden Gleichungen, die zwar zum Teil schon in diesem Abschnitt angesprochen wurden, auf die aber in den nächsten Abschnitten, insbesondere im Abschnitt 2.4 über Estimierungsverfahren, ausgiebig zurückgegriffen werden wird:

$$\begin{aligned}
\text{argmax}_p\ \ p(f) &= \text{argmax}_p \prod_{y\in\mathcal{Y}} p(y)^{f(y)} \\
&= \text{argmax}_p\ \ \tilde{p}[\log\ p] \\
&= \text{argmax}_p\ \ L(p;\tilde{p}) \\
&= \text{argmin}_{\,p}\ H_{\text{cross}}(p;\tilde{p}) \\
&= \text{argmin}_{\,p}\ D(\tilde{p}\|p) \\
&= \text{argmin}_{\,p}\ \text{perplexity}(p;\tilde{p}) \\
&= \text{argmin}_{\,p}\ \text{perplexity}_D(p;\tilde{p}) \ .
\end{aligned}$$

2.4 Estimierungsverfahren und empirische Wahrscheinlichkeiten

In Abschnitt 2.1 wurde ausgeführt, dass der wesentliche Aspekt einer Wahrscheinlichkeitsverteilung darin besteht, dass sie für jeden Datentyp die Information bereithält, mit welcher Wahrscheinlichkeit dieser Datentyp aus einer Grundmenge von Datentypen heraus erzeugt wird. Ferner wurde erwähnt, dass mit diesem Aspekt die Frage, wie oft ein linguistischer Datentyp in einem gegebenem Korpus vorkommt, sehr eng verbunden ist. In diesem Abschnitt wird angerissen, wie diese Verbindung möglicherweise aussehen könnte. Die vorsichtige Wortwahl soll darauf hinweisen, dass sich die Stochastiker in der Beantwortung dieser Frage leider uneinig sind.

Unabhängig von dieser Uneinigkeit der Stochastiker ist die Problemstellung die folgende:

Sei $\mathcal{Y}$ eine Grundmenge von linguistischen Datentypen, und $f(.)$ ein Korpus. Sei $\{p_\Theta(.) \mid \Theta \in \Omega\}$ eine Menge von *parametrisierten Wahrscheinlichkeitsverteilungen* über der gegebenen linguistischen Grundmenge $\mathcal{Y}$, wobei in diesem Zusammenhang eine Menge Ω *Parameterraum* und die $\Theta \in \Omega$ *Parameter* genannt werden. Im allgemeinen kann man sich die Parameter als hochdimensionale reellwertige Vektoren vorstellen, die in der Regel einer Reihe von simplen Nebenbedingungen unterworfen sind.

Die entscheidende Frage ist nun: unter der Vermutung, dass der Korpus mit irgendeiner der vorgegebenen Wahrscheinlichkeitsverteilungen $p_\Theta(.)$ generiert wurde, welcher der in Frage kommenden Parameter $\Theta \in \Omega$ ist der *wahre Parameter* $\hat{\Theta}$, oder anders gefragt: mit welchem expliziten Parameter $\hat{\Theta}$ wurde der Korpus vermutlich erzeugt?

Zur Beantwortung dieser Frage, d.h. zur Estimierung des fraglichen Parameters, stehen offensichtlich nur die gegebene Häufigkeitsverteilung$f(.)$ aller Datentypen im Korpus und die möglichen Wahrscheinlichkeitsverteilungen $p_\Theta(.), \Theta \in \Omega$, zur Verfügung. Das am häufigsten angewandte Estimierungsverfahren der Statistik ist die sogenannte *Maximum-Likelihood-Estimierung*. Der Name deutet an, dass die Maximum-Likelihood-Estimierung unter den in Frage kommenden Parametern $\Theta \in \Omega$ gerade den Parameter $\hat{\Theta}$ auswählt, der die Korpuswahrscheinlichkeit $p_\Theta(f)$ maximiert:

$$\hat{\Theta} = \text{argmax}_{\Theta \in \Omega} \quad p_\Theta(f) \ .$$

Obwohl die Maximum-Likelihood-Estimierung unter den gegebenen Bedingungen intuitiv einleuchtend ist, und sie einige hervorragende mathematische Eigenschaften besitzt (sie ist eine konsistente, für grosse Korpora annähernd normalverteilte Estimierung, mit dem wahren Parameter als Erwartungswert und der inversen Fisher-Information als Varianz), ist sie doch nicht konkurrenzlos (Existenz und Eindeutigkeit der Maximum-Likelihood-

Estimierung sind problematisch).

In DeGroot (1989) werden neben der Maximum-Likelihood-Estimierung wenigstens vier weitere Estimierungsverfahren angegeben: die Bayes-Estimierung, die Maximum-A-Posteriori-Estimierung, die Mean-Squared-Error-Estimierung, sowie eine auf Konfidenzintervallen beruhende Estimierung. Hierbei wird jede Abbildung von der Menge $\mathcal{F}_n(\mathcal{Y})$ aller Korpora einer festen Länge n auf den Parameterraum Ω eine *Estimierung* genannt:

$$\delta : \ \mathcal{F}_n(\mathcal{Y}) \longrightarrow \Omega, \quad f \mapsto \delta(f) \ .$$

Paradoxerweise teilt sich die Gemeinschaft der Stochastiker an der Frage der Maximum-A-Posteriori-Estimierung. Paradox ist dies aus mehreren Gründen. Zunächst einmal deswegen, weil die Maximum-A-Posteriori-Estimierung ohne ein zusätzliches Inventar überhaupt nicht auf das obige Estimierungsproblem angewandt werden kann. Bei diesem zusätzlichen Inventar handelt es sich um die sogenannte *A-Priori-Wahrscheinlichkeitsverteilung*, eine (in der Regel nicht-diskrete) Wahrscheinlichkeitsverteilung auf dem Parameterraum:

$$p : \ \Omega \to [0,1], \quad \Theta \mapsto p(\Theta) \ .$$

Anstatt wie bei der Maximum-Likelihood-Estimierung die Korpuswahrscheinlichkeit $p_\Theta(f)$ zu maximieren, wird bei der *Maximum-A-Posteriori-Estimierung* das Produkt aus A-Priori-Wahrscheinlichkeit und Korpuswahrscheinlichkeit maximiert:

$$\hat{\Theta} = \text{argmax}_{\Theta \in \Omega} \quad p(\Theta) \cdot p_\Theta(f) \ .$$

Da die Produkt-Wahrscheinlichkeit

$$p(\Theta, f) := p(\Theta) \cdot p_\Theta(f)$$

eine (in der Regel nicht-diskrete) parametrisierte Wahrscheinlichkeitsverteilung auf dem Produkt-Raum $\Omega \times \mathcal{F}_n(\mathcal{Y})$ ist:

$$\sum_{\langle \Theta, f \rangle \in \Omega \times \mathcal{F}_n(\mathcal{Y})} p(\Theta, f) = \sum_{\Theta \in \Omega} p(\Theta) \sum_{f \in \mathcal{F}_n(\mathcal{Y})} p_\Theta(f) = \sum_{\Theta \in \Omega} p(\Theta) \cdot 1 = 1 \ ,$$

kann die Maximum-A-Posteriori-Estimierung aus formaler Sicht als eine Maximum-Likelihood-Estimierung interpretiert werden (was tatsächlich nur bedeutet, dass ein Parameter mittels des Kriteriums der Maximierung einer parametrisierten Wahrscheinlichkeit ausgewählt wird). Umgekehrt kann eine Maximum-Likelihood-Estimierung offensichtlich als eine Maximum-A-Posteriori-Estimierung mit gleichverteilter/uniformer A-Priori-Wahrscheinlichkeitsverteilung angesehen werden.

Trotz der formalen Gleichrangigkeit von Maximum-A-Posteriori-Estimierung und Maximum-Likelihood-Estimierung entbrannte unter den Stochastikern ein Streit um die Akzeptanz der Maximum-A-Posteriori-Estimierung mit mehr oder weniger trivialem Hintergrund: Es scheint klar zu sein, dass eine "A-Priori-Bevorzugung" eines Parameters eine A-Posteriori-Estimierung genau dieses Parameters provoziert. Die Frage ist nun, ob deshalb eine A-Priori-Wahrscheinlichkeitsverteilung in eine Parameter-Estimierung einfliessen darf? Wenn ja, gibt es eine Methode mit der die A-Priori-Wahrscheinlichkeitsverteilung aus objektiven Wissensquellen gewonnen werden kann? Obwohl die Diskussion um das beste Estimierungsverfahren sicher fruchtbar ist, geht sie doch über den Rahmen dieser Arbeit weit hinaus. So soll an dieser Stelle nur darauf hingewiesen werden, dass in dieser Arbeit der EM-Algorithmus auf Grundlage der Maximum-Likelihood-Estimierung vorgestellt wird, dass aber einige Varianten des EM-Algorithmus, zum Beispiel sogenanntes "Penalized EM", auf der Maximum-A-Posteriori-Estimierung beruhen.

Im folgenden soll ein Beispiel vorgestellt werden, mit dem zur einleitenden Bemerkung zurückgekehrt wird: Der wesentliche Aspekt einer Wahrscheinlichkeitsverteilung besteht darin, dass sie für jeden Datentyp die Information bereithält, mit welcher Wahrscheinlichkeit dieser Datentyp aus einer Grundmenge von Datentypen heraus erzeugt wird. Es soll nun gezeigt werden, dass hiermit die Frage, wie oft ein linguistischer Datentyp in einem gegebenen Korpus vorkommt, sehr eng verbunden ist. Das Problem wird angegangen, indem von einem gegebenen Korpus $f(.)$ von Datentypen aus $\mathcal{Y}$ ausgegangen wird. Die zu beantwortende Frage ist, welche Wahrscheinlichkeitsverteilung $p(.)$ auf $\mathcal{Y}$ diesen Korpus angemessen repräsentiert. Der Abschnitt 2.1 legt nahe, dass dies die empirische Wahrscheinlichkeitsverteilung $\tilde{p}(.)$ sein wird. Mit der Maximum-Likelihood-Estimierung soll dies nun bewiesen werden. Die Maximum-Likelihood-Estimierung hat auf die Frage der "angemessenen Repräsentation" die Antwort, dass es diejenige Wahrscheinlichkeitsverteilung ist, welche die Korpuswahrscheinlichkeit maximiert:

$$\hat{p} = \text{argmax}_p \quad p(f) \ .$$

Der Abschnitt 2.3 gibt weitere Antworten: Die Maximierung der Korpuswahrscheinlichkeit ist äquivalent zur Maximierung der Log-Likelihood $L(p;\tilde{p})$ oder zur Minimierung der Cross-Entropie $H_{\text{cross}}(p;\tilde{p})$, oder der Kullback-Leibler-Distanz $D(\tilde{p}\|p)$ oder der Korpusperplexität $\text{perplexity}(p,\tilde{p})$. In der Regel wird für das weitere Vorgehen die Maximierung der Log-Likelihood ausgewählt:

$$\begin{aligned} \hat{p} &= \text{argmax}_p \quad L(p;\tilde{p}) \\ &= \text{argmax}_p \quad \tilde{p}[\log\ p] \\ &= \text{argmax}_p \quad \sum_{y\in\mathcal{Y}} \tilde{p}(y) \cdot \log\ p(y) \end{aligned}$$

Zur Lösung dieses *Optimierungsproblems* ist es zweckmässig, eine *Parametrisierung* der Wahrscheinlichkeitsverteilung $p(.)$ vorzunehmen:

$$p(y) := \Theta_y \qquad \forall y \in \mathcal{Y} \ .$$

Weil $p(.)$ eine beliebige Wahrscheinlichkeitsverteilung ist, kann über die *Parameter*

$$\Theta_y \in [0,1]$$

nichts weiter ausgesagt werden, als dass der *Parametervektor*

$$\Theta :=< \Theta_y \mid y \in \mathcal{Y} >$$

ein Element des folgenden *Parameterraums* ist:

$$\Omega := \left\{ \Theta \in [0,1]^{|\mathcal{Y}|} \;\middle|\; g(\Theta) = 0 \right\} \ .$$

Die *Nebenbedingung* in der Definition des Parameterraums, mit

$$g(\Theta) := 1 - \sum_{y \in \mathcal{Y}} \Theta_y \ ,$$

ist offenbar nötig, damit die Parametrisierung für jede Wahl von $\Theta \in \Omega$ eine Wahrscheinlichkeitsverteilung $p(.)$ auf $\mathcal{Y}$ liefert:

$$\sum_{y \in \mathcal{Y}} p(y) = \sum_{y \in \mathcal{Y}} \Theta_y = 1 - g(\Theta) = 1 - 0 = 1 \ .$$

Weitere Nebenbedingungen würden den Parameterraum und damit die Menge der parametrisierten Wahrscheinlichkeitsverteilungen weiter einschränken; was für andere, als die hier zu lösende Fragestellung, durchaus sinnvoll sein kann.

Offensichtlich ist die Wahrscheinlichkeitsverteilung $p(.)$ durch die vorgenommene Parametrisierung von dem Parametervektor Θ abhängig geworden, was wie folgt notiert werden soll:

$$p(.) = p_\Theta(.) \ .$$

Dasselbe gilt dann aber auch für die Log-Likelihood, was ebenfalls explizit notiert wird:

$$L(\Theta) := L(p_\Theta; \tilde{p}) \ .$$

Leider hat das Auftreten der Nebenbedingung $g(\Theta) = 0$ zur Folge, dass das zu lösende Optimierungsproblem, welches sich mittlerweise in der Form:

$$\begin{aligned} \hat{\Theta} &= \text{argmax}_{\Theta \in \Omega} \quad L(\Theta) \\ &= \text{argmax}_{\Theta \in \Omega} \quad \sum_{y \in \mathcal{Y}} \tilde{p}(y) \cdot \log \ p_\Theta(y) \\ &= \text{argmax}_{\Theta \in \Omega} \quad \sum_{y \in \mathcal{Y}} \tilde{p}(y) \cdot \log \ \Theta_y \end{aligned}$$

präsentiert, nur mit einem tiefen Satz aus der *Differentialrechnung mehrerer Veränderlichen* angegangen werden kann. Im folgenden Satz von J.L. Lagrange, der als sogenannte *Multiplikatorregel* bekannt ist (Erwe 1962), wird eine **notwendige** Bedingung für das Vorliegen eines sogenannten *Extremum mit Nebenbedingungen* angegeben:

Theorem (Multiplikatorregel von J.L. Lagrange): Seien $L(\Theta)$ und $g_1(\Theta) \ldots g_N(\Theta)$ in einer Umgebung eines Punktes $\hat{\Theta} \in [0,1]^{|\mathcal{Y}|}$ stetig differenzierbare reellwertige Abbildungen ($N < |\mathcal{Y}|$). Die *Funktionalmatrix*:

$$\begin{pmatrix} \nabla_\Theta\ g_1 \\ \vdots \\ \nabla_\Theta\ g_N \end{pmatrix}$$

besitze am Punkt $\hat{\Theta}$ maximalen Rang, während

$$g_1(\hat{\Theta}) = \ldots = g_N(\hat{\Theta}) = 0$$

ist. Wenn dann $L(\hat{\Theta})$ Extremum der Werte von $L(\Theta)$ bei Beschränkung auf solche Θ ist, für die:

$$g_1(\Theta) = \ldots = g_N(\Theta) = 0$$

ist, dann gibt es Zahlen $\lambda_1 \ldots \lambda_N$, sodass in $\hat{\Theta}$ gilt:

$$\frac{\partial\ L}{\partial\ \Theta_y} + \lambda_1 \cdot \frac{\partial\ g_1}{\partial\ \Theta_y} + \ldots \lambda_N \cdot \frac{\partial\ g_N}{\partial\ \Theta_y} \ = \ 0 \qquad\qquad \forall y \in \mathcal{Y}\ .$$

Bemerkung: Der *Gradient* ∇_Θ ist der Vektor der *ersten partiellen Ableitungen* einer Funktion nach allen ihren Parameter-Argumenten:

$$\nabla_\Theta\ L := \left\langle \frac{\partial\ L}{\partial\ \Theta_y} \middle| y \in \mathcal{Y} \right\rangle\ .$$

Um die Extrema einer Funktion $L(\Theta)$ unter den Nebenbedingungen $g_1(\Theta) = \ldots = g_N(\Theta) = 0$ zu ermitteln, hat man wie folgt vorzugehen: Von der Funktion $L(\Theta)$ geht man mittels noch unbekannter reeller Zahlen $\lambda_1 \ldots \lambda_N$ (sogenannten *Multiplikatoren*) zu der sogenannten *Lagrangefunktion*:

$$\Lambda(\Theta) := L(\Theta) + \lambda_1 \cdot g_1(\Theta) + \ldots \lambda_N \cdot g_N(\Theta)$$

über und tut so, als müsste man deren Extrema bestimmen, dann führt dies zu den in der Multiplikatorregel angegebenen $|\mathcal{Y}|$ Gleichungen. Diese ergeben zusammen mit den N Nebenbedingungen ein System von $|\mathcal{Y}| + N$ Gleichungen für die $|\mathcal{Y}| + N$ Unbekannten $\Theta_y, (y \in \mathcal{Y})$ und $\lambda_1 \ldots \lambda_N$. Höchstens die als Lösungen dieses Gleichungssystems auftretenden Θ, kommen als Stellen in Frage, in denen $L(.)$ ein Extremum mit jenen Nebenbedingungen besitzt. Eine solche Stelle braucht allerdings nicht stets wirklich ein derartiges

Extremum zu liefern, nicht einmal ein "relatives". Die Methode wird besonders in denjenigen Fällen nützlich sein, in denen aus anderen Gründen die Existenz eines Extremums sichergestellt ist und nur noch die Berechnung aussteht.

Die Anwendung der Multiplikatorregel auf die Log-Likelihood ergibt offenbar die folgende Lagrangefunktion ($N = 1$):

$$\begin{aligned}\Lambda(\Theta) &:= L(\Theta) + \lambda \cdot g(\Theta) \\ &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \cdot \log\ \Theta_y + \lambda \cdot \left(1 - \sum_{y \in \mathcal{Y}} \Theta_y\right) \ .\end{aligned}$$

Die ersten partiellen Ableitungen sind ($y \in \mathcal{Y}$):

$$\frac{\partial\ \Lambda}{\partial\ \Theta_y} := \frac{\tilde{p}(y)}{\Theta_y} - \lambda \ .$$

Das Gleichungssystem aus der Multiplikatorregel erhält man, indem die ersten partiellen Ableitungen der Lagrangefunktion gleich Null gesetzt werden ($y \in \mathcal{Y}$):

$$0 = \frac{\tilde{p}(y)}{\hat{\Theta}_y} - \lambda \ .$$

Dies liefert $\hat{\Theta}_y = \frac{\tilde{p}(y)}{\lambda}$. Die Nebenbedingung liefert noch, dass $\lambda = 1$ ist, d.h. es gilt $\hat{\Theta}_y = \tilde{p}(y)$. Zusammenfassend erhält man die folgende mögliche Lösung der Ausgangsfragestellung:

$$\begin{aligned}\hat{p}(y) &= p_{\hat{\Theta}}(y) \\ &= \hat{\Theta}_y \\ &= \tilde{p}(y) \ .\end{aligned}$$

D.h. die Vermutung erhärtet sich, dass die empirische Wahrscheinlichkeitsverteilung diejenige Wahrscheinlichkeitsverteilung ist, die die Korpuswahrscheinlichkeit maximiert und damit den Korpus am angemessensten repräsentiert. Leider ist die Bemerkung, die sich der Multiplikatorregel anschliesst, sehr ernst zu nehmen: Parameter, die mit der Multiplikatorregel errechnet werden, können Extrema sein, müssen es aber nicht sein; hierfür gibt es in der Literatur eine reiche Anzahl von Beispielen. Zudem weiss man (ohne zusätzliche Berechnungen) nicht, welche Art von Extremum, Minimum oder Maximum, berechnet wurde. Deshalb soll im folgenden ein alternatives Verfahren vorgestellt werden, mit dem man in die Lage versetzt wird, die Extrema der Log-Likelihood zu berechnen, zusätzlich aber auch nachprüfen kann, ob ein Minimum oder Maximum vorliegt. Leider ist auch für dieses Verfahren ein tiefer Satz aus der Differentialrechnung mehrerer Veränderlichen nötig. Er gibt hinreichende Bedingungen für das Vorliegen von Extrema an und wurde wieder Erwe (1962) entnommen.

Theorem: Wenn $L(\Theta)$ zweimal stetig differenzierbar nach allen ihren Variablen ist und in $\hat{\Theta}$

$$\nabla_\Theta L = 0$$

gilt, und die Matrix

$$\left(\frac{\partial^2 L}{\partial \Theta_y \partial \Theta_{y'}} \right)_{y,y' \in \mathcal{Y}}$$

in $\hat{\Theta}$ *negativ definit* ist, so hat $L(\Theta)$ in $\hat{\Theta}$ ein ***strenges*** *relatives Maximum*. Ist die Matrix hingegen *positiv definit*, so hat $L(\Theta)$ in $\hat{\Theta}$ ein ***strenges*** *relatives Minimum*.

Bemerkung: Man sagt, $L(\Theta)$ besitze in $\hat{\Theta}$ ein ***strenges*** *relatives Maximum*, wenn es eine Umgebung $\mathcal{U}$ von $\hat{\Theta}$ gibt, sodass:

$$L(\Theta) < L(\hat{\Theta}) \qquad \forall \Theta \in \mathcal{U} \ .$$

Entsprechendes gilt für ein ***strenges*** *relatives Minimum*. Man sagt ferner, eine *symmetrische Matrix*

$$\mathcal{M} = \left(m_{\ \Theta_y \Theta_{y'}} \right)_{y,y' \in \mathcal{Y}}$$

ist *positiv definit*, falls für beliebige reelwertige Vektoren $\Theta \neq 0$ gilt, dass:

$$\Theta^t \cdot \mathcal{M} \cdot \Theta \ > \ 0 \ .$$

Entsprechendes gilt für *negativ definite* Matrizen.

Im folgenden wird vorgestellt, wie dieses Theorem benutzt werden kann, um zu zeigen, dass die empirische Verteilung $\tilde{p}(.)$ tatsächlich die Korpuswahrscheinlichkeit maximiert. Die geradlinigste Vorgehensweise ist, eine Parametrisierung herzustellen, die ohne Nebenbedingungen auskommt. Mit einer Parametrisierung ohne Nebenbedingungen sollte es möglich sein, mit dem ersten Teil des Theorems, mögliche Extrema zu berechnen. Mit dem zweiten Teil des Theorems sollte verifizierbar sein, welche der möglichen Extrema tatsächlich Maxima sind. Die skizzierte Vorgehensweise wird erfolgreich sein. Obwohl das gesamte Verfahren sehr einfach ist, ist dem Autor nicht bekannt, ob es schon einmal irgendwo in der Literatur angewandt wurde. In Abschnitt 3.8 wird es im Rahmen des EM-Algorithmus erneut für kontextfreie Grammatiken benutzt werden.

Der Schlüsselschritt ist, eine Parametrisierung zu konstruieren, die ohne Nebenbedingungen auskommt. Hierfür wird in der Menge $\mathcal{Y}$ der Datentypen zunächst ein beliebiger Datentyp y_0 markiert, und danach $\mathcal{Y}$ wie folgt aufgespaltet:

$$\mathcal{Y} = \{y_0\} + \mathcal{Y}_0 \ .$$

(Das '+'-Zeichen steht für die disjunkte Vereinigung von Mengen.) Die gewünschte Parametrisierung wird folgendermassen vorgenommen:

$$p_\Theta(y) := \begin{cases} 1 - \sum_{y \in \mathcal{Y}_0} \Theta_y & \text{falls } y = y_0 \\ \Theta_y & \text{falls } y \in \mathcal{Y}_0 \end{cases}$$

Unten wird sich zeigen, dass ausserdem $\tilde{p}(y_0) \neq 0$ gelten muss. Da der Korpus nicht leer ist, muss es natürlich einen solchen Datentyp geben. Die Parametrisierung kommt mit $|\mathcal{Y}_0| = |\mathcal{Y}| - 1$ Parametern Θ_y, $(y \in \mathcal{Y}_0)$ aus. Für die neue Parametrisierung wird im Vergleich zu der alten Parametrisierung also ein Parameter weniger benötigt. Ferner ist für sie keine Nebenbedingung erforderlich, weil:

$$\begin{aligned} \sum_{y \in \mathcal{Y}} p_\Theta(y) &= p_\Theta(y_0) + \sum_{y \in \mathcal{Y}_0} p_\Theta(y) \\ &= \left(1 - \sum_{y \in \mathcal{Y}_0} \Theta_y\right) + \sum_{y \in \mathcal{Y}_0} \Theta_y \\ &= 1 \ . \end{aligned}$$

Der Parameterraum ist daher von der einfachen Gestalt[13]:

$$\Omega_0 := [0,1]^{|\mathcal{Y}_0|} \ .$$

Die Log-Likelihood ergibt sich für diese Parametrisierung, wie folgt:

$$\begin{aligned} L_0(\Theta) &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \cdot \log\ p_\Theta(y) \\ &= \tilde{p}(y_0) \cdot \log\ p_\Theta(y_0) + \sum_{y \in \mathcal{Y}_0} \tilde{p}(y) \cdot \log\ p_\Theta(y) \\ &= \tilde{p}(y_0) \cdot \log \left(1 - \sum_{y \in \mathcal{Y}_0} \Theta_y\right) + \sum_{y \in \mathcal{Y}_0} \tilde{p}(y) \cdot \log\ \Theta_y \ . \end{aligned}$$

Vergleicht man die neue Log-Likelihood $L_0(\Theta)$ mit der Lagrangefunktion $\Lambda(\Theta)$ für die alte Parametrisierung, so fällt die grosse Ähnlichkeit dieser beiden Funktionen auf: Der wesentliche Unterschied ist, dass $L_0(\Theta)$ einen Term $\log\left(1 - \sum_y \Theta_y\right)$ enthält, und $\Lambda(\Theta)$ den einfacheren Term $\left(1 - \sum_y \Theta_y\right)$. Es wird sich im folgenden zeigen, dass dieser Unterschied nur zu einem unwesentlich grösseren Aufwand zur Berechnung der Extrema führt[14].

[13] Natürlich muss zusätzlich gewährleistet sein, dass

$$0 \leq p_\Theta(y_0) = 1 - \sum_{y \in \mathcal{Y}_0} \Theta_y \leq 1 \ .$$

Diese Nebenbedingung kann jedoch nach der Optimierung der Parameter überprüft werden und wird dann in den meisten Fällen erfüllt sein. Im Gegensatz hierzu hätte für die alte Parametrisierung eine Optimierung ohne Nebenbedingung keinesfalls Parameter geliefert, die im nachhinein $g(\Theta) = 0$ erfüllt hätten.

[14] Der zusätzliche Rechenaufwand, der sich im Vergleich zur alten Parametrisierung ergibt, wird geleistet, um zu zeigen, dass die errechneten Extrema Maxima sind; was mit der alten Parametrisierung prinzipiell unmöglich ist.

Der erste Teil des Theorems verlangt, dass der Gradient der Log-Likelihood $L_0(\Theta)$ am Extremum gleich Null ist. Die ersten partiellen Ableitungen von $L_0(\Theta)$ sind ($y \in \mathcal{Y}_0$):

$$\frac{\partial\, L_0}{\partial\, \Theta_y} \;=\; \tilde{p}(y_0) \cdot \frac{-1}{\left(1 - \sum_{y \in \mathcal{Y}_0} \Theta_y\right)} + \tilde{p}(y) \cdot \frac{1}{\Theta_y} \;.$$

Setzt man diese gleich Null, erhält man das folgende lineare Gleichungssystem mit $|\mathcal{Y}_0|$ Gleichungen und $|\mathcal{Y}_0|$ Unbekannten ($y \in \mathcal{Y}_0$):

$$\left(1 - \sum_{y \in \mathcal{Y}_0} \Theta_y\right) \cdot \tilde{p}(y) = \tilde{p}(y_0) \cdot \Theta_y \;.$$

Man entnimmt den Gleichungen, dass sich Θ_y und $\tilde{p}(y)$ proportional zueinander verhalten; die einzige Lösung ist daher ($y \in \mathcal{Y}_0$):

$$\hat{\Theta}_y = \tilde{p}(y) \;.$$

Der zweite Teil des Theorems verlangt die Berechnung der Matrix:

$$\left(\frac{\partial^2 L_0}{\partial \Theta_y \partial \Theta_{y'}}\right)_{y,y' \in \mathcal{Y}_0}$$

an der Stelle $\hat{\Theta}$. Für $y, y' \in \mathcal{Y}_0$ gilt:

$$\begin{aligned}
\frac{\partial^2 L_0}{\partial \Theta_y \partial \Theta_{y'}} &= \frac{\partial}{\partial\, \Theta_y} \left(\tilde{p}(y_0) \cdot \frac{-1}{\left(1 - \sum_{y \in \mathcal{Y}_0} \Theta_y\right)} + \tilde{p}(y') \cdot \frac{1}{\Theta_{y'}} \right) \\
&= \tilde{p}(y_0) \cdot \frac{-1}{\left(1 - \sum_{y \in \mathcal{Y}_0} \Theta_y\right)^2} + \begin{cases} 0 & \text{falls } y \neq y' \\ -\tilde{p}(y) \cdot \frac{1}{\Theta_y^2} & \text{falls } y = y' \end{cases}
\end{aligned}$$

An der Stelle $\hat{\Theta}$ erhält man daher:

$$\begin{aligned}
\left.\frac{\partial^2 L_0}{\partial \Theta_y \partial \Theta_{y'}}\right|_{\hat{\Theta} = \tilde{p}} &= \tilde{p}(y_0) \cdot \frac{-1}{\left(1 - \sum_{y \in \mathcal{Y}_0} \tilde{p}(y)\right)^2} + \begin{cases} 0 & \text{falls } y \neq y' \\ -\tilde{p}(y) \cdot \frac{1}{\tilde{p}(y)^2} & \text{falls } y = y' \end{cases} \\
&= \begin{cases} -\frac{1}{\tilde{p}(y_0)} & \text{falls } y \neq y' \\ -\frac{1}{\tilde{p}(y_0)} - \frac{1}{\tilde{p}(y)} & \text{falls } y = y' \end{cases}
\end{aligned}$$

Die Matrix:

$$\left(\frac{\partial^2 L_0}{\partial \Theta_y \partial \Theta_{y'}}\right)_{y,y' \in \mathcal{Y}_0}$$

lässt sich an der Stelle $\hat{\Theta}$ also als Summe $\mathcal{C} + \mathcal{D}$ zweier Matrizen $\mathcal{C}$ und $\mathcal{D}$ darstellen, wobei die Matrix $\mathcal{C}$ eine *konstante Matrix* mit den Matrizenelementen $-\frac{1}{\tilde{p}(y_0)}$, und die Matrix $\mathcal{D}$

eine *Diagonalmatrix* mit den Diagonalelementen $-\frac{1}{\tilde{p}(y)}$, $(y \in \mathcal{Y}_0)$ ist. Im folgenden wird gezeigt, dass beide Matrizen $\mathcal{C}$ und $\mathcal{D}$ negativ definit sind. Für beliebige reellwertige Vektoren $\Theta \neq 0$ gilt:

$$\begin{aligned}
\Theta^t \cdot \mathcal{C} \cdot \Theta &= \left(\sum_{y \in \mathcal{Y}_0} \Theta_y\right) \cdot \frac{-1}{\tilde{p}(y_0)} \cdot \left(\sum_{y \in \mathcal{Y}_0} \Theta_y\right) \\
&= -\frac{1}{\tilde{p}(y_0)} \cdot \left(\sum_{y \in \mathcal{Y}_0} \Theta_y\right)^2 \\
&< 0 \ ,
\end{aligned}$$

vorausgesetzt, dass die empirische Wahrscheinlichkeit des markierten Elements nicht gleich Null ist. Solch ein Element muss aber existieren, denn sonst wäre der Korpus leer. Ferner gilt:

$$\begin{aligned}
\Theta^t \cdot \mathcal{D} \cdot \Theta &= \sum_{y \in \mathcal{Y}_0} \Theta_y \cdot \left(-\frac{1}{\tilde{p}(y_0)} - \frac{1}{\tilde{p}(y)}\right) \cdot \Theta_y \\
&= \sum_{y \in \mathcal{Y}_0} \left(-\frac{1}{\tilde{p}(y_0)} - \frac{1}{\tilde{p}(y)}\right) \cdot \Theta_y^2 \\
&< 0 \ ,
\end{aligned}$$

mit derselben Begründung. Es folgt, dass sowohl die Matrix $\mathcal{C}$ als auch die Matrix $\mathcal{D}$ negativ definit ist. Da die Summe zweier negativ definiter Matrizen ebenfalls negativ definit ist, ist alles gezeigt. Gemäss Theorem folgt, dass:

$$\hat{\Theta}_y = \tilde{p}(y), \qquad (y \in \mathcal{Y}_0),$$

ein strenges relatives Maximum der Log-Likelihood $L_0(\Theta)$ ist.

Da sehr einfach zu zeigen ist, dass $L_0(\Theta)$ auf dem *Rand* des Parameterraums $\Omega_0 = [0,1]^{|\mathcal{Y}_0|}$ nur den Wert $-\infty$ annimmt, welcher nicht grösser als $L_0(\hat{\Theta})$ ist, ist $\hat{\Theta} = \tilde{p}$ sogar ein *globales Maximum* der Log-Likelihood $L_0(\Theta)$.

Damit konnte das angekündigte Ziel bewiesen werden, dass die empirische Wahrscheinlichkeitsverteilung einen Korpus am besten repräsentiert: Unter allen Wahrscheinlichkeitsverteilungen ist es die empirische Wahrscheinlichkeitsverteilung, welche die Log-Likelihood (bzw. die Korpuswahrscheinlichkeit) maximiert:

$$\tilde{p} = \text{argmax}_p \ \ L(p; \tilde{p}) \ .$$

Dies ist äquivalent zu der Aussage, dass für alle Wahrscheinlichkeitsverteilungen $p(.)$ gilt, dass: $L(\tilde{p}; \tilde{p}) \geq L(p; \tilde{p})$. Es ist recht interessant, dass dieses befriedigende Ergebnis zeigt (gleichwertig zu der Aussage ist), dass die Cross-Entropie stets grösser als die Entropie ist. Mit $H(\tilde{p}) = -L(\tilde{p}; \tilde{p})$ und $H_{\text{cross}}(p; \tilde{p}) = -L(p; \tilde{p})$ folgt nämlich, dass:

$$H_{\text{cross}}(p; \tilde{p}) \geq H(\tilde{p}) \qquad \forall p \ .$$

Im Abschnitt 2.3 wurde darauf hingewiesen, dass diese Ungleichung kein triviales Resultat ist, sondern die fundamentale *Informationsungleichung der Informationstheorie* beweist, welche aussagt, dass die Kullback-Leibler-Distanz $D(\tilde{p}\|p)$ keine negativen Werte annimmt:

$$D(\tilde{p}\|p) = H_{\text{cross}}(p;\tilde{p}) - H(\tilde{p}) \geq 0 \ .$$

Mit diesem schönen Ergebnis soll der Abschnitt beschlossen werden.

2.5 Baumbank-Trainingsverfahren

Das sogenannte *Baumbank-Trainingsverfahren* ist ein einfaches, nicht-iteratives Trainingsverfahren für eine probabilistische kontextfreie Grammatik. Das Baumbank-Trainingsverfahren ist eine Maximum-Likelihood-Estimierung mit annotierten Daten, wobei die Regelwahrscheinlichkeiten der gegebenen kontextfreien Grammatik als Parameter aufgefasst und estimiert werden. Das Baumbank-Trainingsverfahren setzt die Existenz einer sogenannten *Baumbank* voraus, d.h. eines grossen Korpus von manuell annotierten Sätzen. Gemäss der Maximum-Likelihood-Estimierung werden die Regelwahrscheinlichkeiten so gewählt, dass die Wahrscheinlichkeit der Baumbank maximiert wird.

Das Baumbank-Trainingsverfahren hat einige sehr gute Eigenschaften. Der eine Grund ist, dass das Baumbank-Trainingsverfahren als Maximum-Likelihood-Estimierung deren gute stochastische Eigenschaften erbt. Der andere Grund ist, dass die mit dem Baumbank-Trainingsverfahren trainierten Grammatiken in der Regel hervorragende linguistische Eigenschaften besitzen. Die Ursache hierfür ist, dass die Annotationen aus den manuell disambiguierten Syntaxbäumen aller Sätze eines grossen Textkorpus bestehen. Dies hat zwei Vorteile. Erstens ist in den Annotationen ein umfangreiches linguistisches Expertenwissen kodiert, welches direkt in die maschinell von der Baumbank abgelesenen Grammatikregeln zurückfliessen kann. Zweitens können die Vorkommenshäufigkeiten der Grammatikregeln direkt in der gegebenen Baumbank gezählt werden, was besonders gute, an diese Baumbank angepasste, Regelwahrscheinlichkeiten produziert. Beides führt dazu, dass eine linguistische Evaluierung einer so hergestellten probabilistischen Grammatik sehr gute Ergebnisse zeigt, vorausgesetzt, dass der Testkorpus der Baumbank ähnlich ist.

In der Abbildung 2.7 ist das Baumbank-Trainingsverfahren schematisch dargestellt. Ein- und Ausgabe-Daten wurden mit dem Symbol ⛁ dargestellt, Algorithmen hingegen mit dem Symbol ▭. Die Abbildung orientiert sich an der, in Charniak (1996) vorgestellten Vorgehensweise. Interessant daran ist, dass Charniak eine gegebene Baumbank nicht allein zum Training einer gegebenen kontextfreien Grammatik benutzt, sondern in einem vorangehenden Schritt die kontextfreie Grammatik unter Benutzung der gegebenen Baumbank herstellt. Der Vorteil dieser Vorgehensweise liegt klar auf der Hand: eine kontextfreie Grammatik muss nicht erst in langwieriger, manueller und deshalb teurer Entwicklungsarbeit hergestellt werden, sondern kann fast ohne Kosten maschinell direkt von der Baumbank abgelesen werden. Der Aufwand der Annotionsarbeit zur Herstellung der Baumbank wird daher teilweise durch die eingesparte Arbeit für die Grammatikentwicklung kompensiert. Ein zweiter Vorteil ist, dass die so abgeleitete Grammatik besonders gut an die gegebene Baumbank angepasst ist. Im folgenden werden die fünf Algorithmen kurz besprochen:

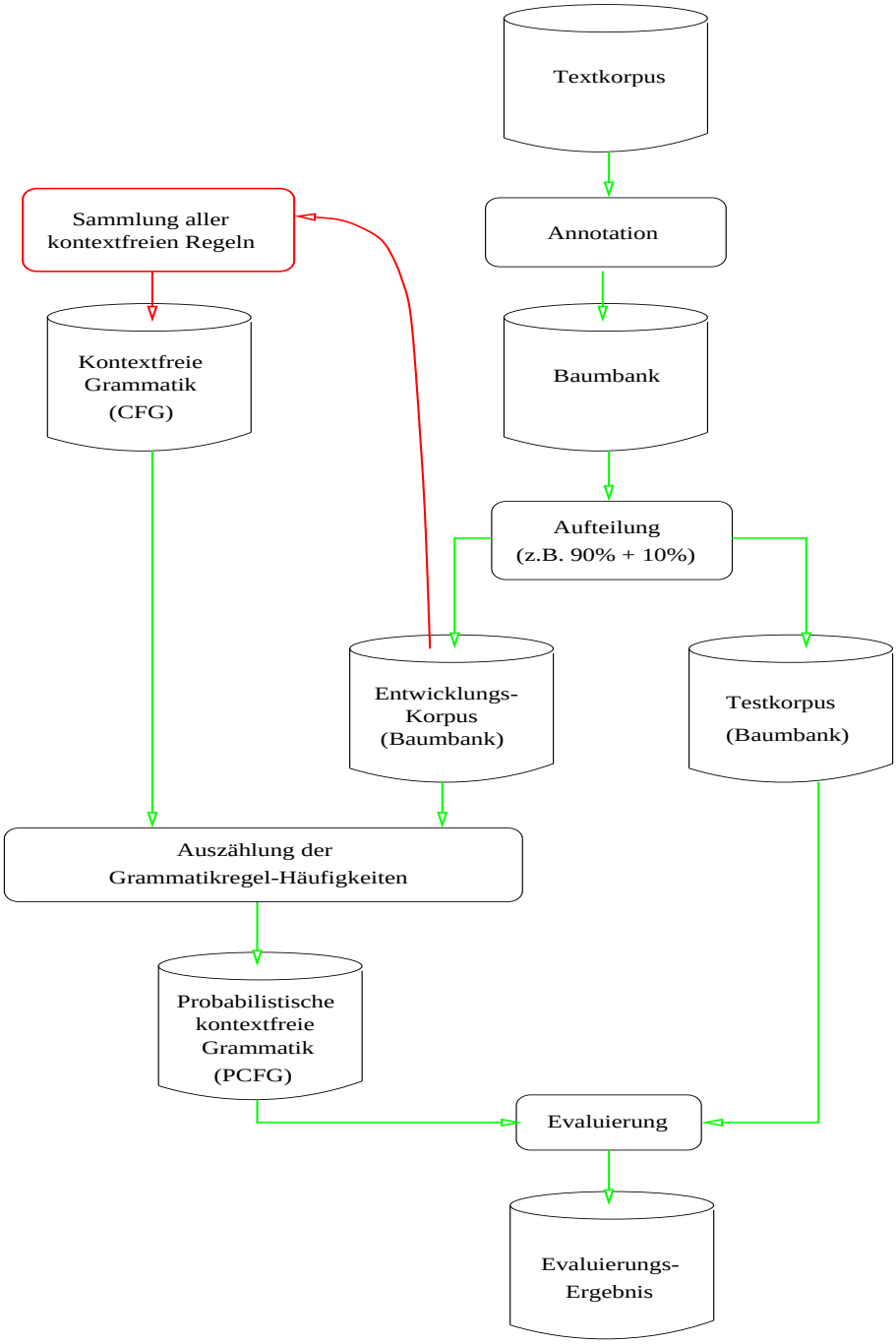


Abbildung 2.7: Herstellung und Training einer kontextfreien Grammatik mit einer Baumbank

- *Annotation* des Textkorpus; das Resultat ist eine *Baumbank*,
- *Aufteilung* der Baumbank in einen *Entwicklungskorpus* und einen *Testkorpus*,
- *Sammlung* aller kontextfreien Regeln des Entwicklungskorpus; das Resultat ist eine *kontextfreie Grammatik*,
- *Auszählung* der Vorkommenshäufigkeiten der Grammatikregeln im Entwicklungskorpus; das Resultat ist eine *probabilistische kontextfreie Grammatik*,
- *Evaluierung* der trainierten probabilistischen kontextfreien Grammatik mit Hilfe des Testkorpus; das Resultat ist ein *Evaluierungsergebnis*.

Die sogenannte *Annotation* eines gegebenen Textkorpus besteht in der Aufgabe, alle Sätze dieses Korpus linguistisch zu analysieren, sodass jeder Satz exakt eine Analyse aufweist. Da diese Aufgabe maschinell nicht in ausreichender Güte vorgenommen werden kann, wird sie in der Regel wenigstens teilweise manuell vorgenommen: Ein menschlicher Annotator hat die Aufgabe, die möglichen Syntaxbäume eines jeden Satzes aus dem Textkorpus (gegebenenfalls mit maschineller Hilfe) zu generieren und zu disambiguieren. Leider ist dies für Sätze mit vielen möglichen Analysen schwer und oft nicht eindeutig lösbar. Daher sind die Annotationen generell nur unter grossem Zeitaufwand zu realisieren. Aus diesem Grund wird der zu annotierende Korpus in der Regel in mehrere Sektionen eingeteilt, sodass die Annotationsarbeit mit parallel zueinander arbeitenden Annotierern geschehen kann. Leider hat dies zur Folge, dass die Qualität der Annotation von den Sektionen stark abhängig ist, sodass Konsistenz oder Homogenität der Baumbank zu einem Problem wird. Zusammenfassend muss man leider konstatieren, dass Baumbanken in der Regel nicht zugleich umfangreich, konsistent und billig sein können. Die wohl am meisten benutzte Baumbank ist die *Penn Wall Street Journal Treebank* (Marcus et al. 1993). Sie ist für Mitglieder des Linguistic Data Consortium (LDC 1993) erhältlich und ist eine Sammlung diverser englisch-sprachiger Texte (ca. 1 Million Wörter).

Die *Aufteilung* der Baumbank in einen *Entwicklungskorpus* und einen *Testkorpus* bereitet den Test vor, wie gut die trainierte Grammatik arbeitet. Damit der Test fair ist, sollten die Testdaten sogenannte *ungesehene Daten* sein. Diese Terminologie bringt zum Ausdruck, dass die Trainingsdaten *gesehene Daten* für ein maschinelles Lernverfahren sind. Natürlich ist es nicht uninteressant, wie ein maschinelles Lernverfahren auf gesehenen Daten arbeitet, d.h. die gesehenen Daten gelernt hat. Die wichtigere Eigenschaft ist aber seine Fähigkeit zur *Generalisierung*, d.h. die Fähigkeit mit Hilfe der gesehenen Daten, neue ungesehene Daten richtig beurteilen zu können. Ein Phänomen, das an dieser Stelle zu nennen ist, ist das sogenannte *Übertraining*. Dies bedeutet, dass ein System auf den gesehenen Daten sehr gute Evaluierungsergebnisse aufweist, jedoch auf den ungesehenen Daten sehr

schlechte. Übertraining erklärt man damit, dass das System ein sehr feines Wissen über die gesehenen Daten gelernt hat, durch eine Überspezialisierung aber leider nicht mehr fähig ist, neue Daten richtig beurteilen zu können[15]. In der Regel wird die Aufteilung der Baumbank in Trainings- und Testdaten so vorgenommen, dass man 90% Trainingsdaten und 10% Testdaten erhält. Natürlich sollten die Testdaten zwar unabhängig von den Trainingsdaten sein, aber diesen auch nicht zu unähnlich sein. Eine übliche Prozedur, um dies zu erreichen, ist es, jeden zehnten Satz der Baumbank als Testdatum zu nehmen. Aus der Penn Wall Street Journal Treebank werden in der Regel die Sektionen 02-21 (ca. 1 Million Wörter) zum Training benutzt und die Sektion 23 (ungefähr 50000 Wörter) zum Testen. Bemerkenswert ist, dass die Testdaten im Gegensatz zu den Trainingsdaten niemals manuell gesichtet werden sollten. Der Grund ist, dass eine manuelle Sichtung der Testdaten dazu führen kann, dass das System während der Entwicklungsphase, unter impliziter Benutzung der Testdaten, dahingehend getunt wird, die Testdaten richtig zu bearbeiten. Eine sicherlich unerwünschte Folge dieses Tunings ist aber, dass das System nicht mehr generalisieren kann. Ein wissenschaftlicher Betrug liegt dann vor, wenn nur getunte Ergebnisse reportiert werden. Daher wird während der Systementwicklung oft ein sogenannter *Entwicklungs-Testkorpus* zur Evaluierung benutzt. Nach vollendeter Entwicklung wird das System abschliessend auf den Testdaten evaluiert. Der Entwicklungs-Testkorpus fehlt in Abbildung 2.7, da in diesem Abschnitt das Baumbank-Trainingsverfahren in seiner einfachsten Form vorgestellt wird, welches von Natur aus kein Tuning zulässt. In späteren Arbeiten weicht Charniak aber von diesem einfachen Baumbank-Trainingsverfahren ab und zieht sehr hilfreiche Wortstatisken (Charniak 1997) hinzu, oder benutzt komplexere Parameter-Estimierungsverfahren (Charniak 2000). Aus der Penn Wall Street Journal Treebank wird in der Regel die Sektion 24 als Entwicklungs-Testkorpus benutzt (Charniak (1997), Collins (1996), Magerman (1995)).

Die Evaluierung einer Grammatik ist in der Regel eine sehr schwierige Aufgabe. Es wäre schön, wenn es für diese Aufgabe eine standardisierte Vorgehensweise gäbe. Dies ist zum Beispiel in der Sprachmodellierung der Fall, in der sich die Korpusperplexität, gemessen auf einem Testkorpus (siehe Abschnitt 2.3), als Standard-Evaluierungsmetrik durchgesetzt hat. Obwohl bekannt ist, dass ein Sprachmodell stets in einer Anwendung getestet werden muss, kursieren "Pi-mal-Daumen"-Regeln der folgenden Art: eine Verbesserung von 5% Korpusperplexität ist für die Anwendung "uninteressant", 10-20% ist "bemerkenswert", und 30% ist "signifikant" (Rosenfeld 2000). Obwohl es alte Ansätze gibt, probabilistische kontextfreie Grammatiken mit Sprachmodellen zu mischen und die Güte dieser hybriden Modelle per Korpusperplexität zu evaluieren (Carroll 1995), werden probabilistische

[15]Das folgende Sprichwort beschreibt Übertraining gut: *Er sieht den Wald vor lauter Bäumen nicht!*

kontextfreie Grammatiken, anders als in der Sprachmodellierung, nicht mittels der Korpusperplexität evaluiert. Die Standard-Vorgehensweise ist, den Testkorpus mit Hilfe der trainierten Grammatik zu parsen und den jeweils wahrscheinlichsten Syntaxbaum (den Viterbi-Parse) mit dem im Testkorpus annotierten Syntaxbaum zu vergleichen. In Abschnitt 2.6 wird gezeigt, wie eine probabilistische kontextfreie Grammatik benutzt werden kann, um Syntaxbäume zu erzeugen und zu disambiguieren. An dieser Stelle soll auf die Vielzahl der Methoden verwiesen werden, mit der zwei Syntaxbäume miteinander verglichen werden. Dem Autor sind die folgenden Evaluierungsmetriken bekannt:

- Die PARSEVAL-Metriken:
  (Labelled) Precision, *(Labelled) Recall*, und *Crossing Brackets*,
- Akkuratheit für *Subkategorisierungsrahmen*,
- Akkuratheit für *Nomen-, Verb-, und Präpositional-Chunks*,
- Akkuratheit für *Exact Match Task*,

Die PARSEVAL-Metriken sind ein Ergebnis der PARSEVAL-Initiative (Black 1992): *Precision* ist der Prozentanteil der *Klammern*[16] im Parse, die auch im annotierten Parse zu finden sind. *Recall* ist der Prozentanteil der Klammern im annotierten Parse, die auch im Parse zu finden sind. *Labelled* heisst, dass hierbei die Grammatikkategorien mit berücksichtigt werden. *Crossing Brackets* ist der Prozentanteil der Klammern, die sich in beiden Parses überschneiden. Diese Masse sind vor allem im Verlauf des noch offenen Wettbewerbs, den besten stochastischen Parser für die Penn Wallstreet Journal Treebank zu bauen, weiter verfeinert worden. Auf der Homepage von Collins und Sekine (2001) kann Software für eine Evaluierung mit den verfeinerten PARSEVAL-Metriken heruntergeladen werden. Die besten stochastischen Parser liefern für alle Metriken weit über 90% Precision/Recall. Akkuratheit für *Subkategorisierungsrahmen* und für *Nomen-, Verb-, und Präpositional-Chunks* misst den Prozentanteil der korrekt erkannten Subkategorisierungsrahmen und Chunks (Abney 1996) im Parse. Evaluierungsergebnisse guter stochastischer Parser liegen typischerweise im Bereich von 80-90% (Beil et al. 1999). Der *Exact Match Task* ist ein sehr hartes Mass, weil es die völlige Übereinstimmung beider Analysen verlangt (Johnson et al. 1999). Soweit dem Autor bekannt ist, wurde es bisher nicht für kontextfreie Grammatiken mit im Durchschnitt vielen tausend Analysen, sondern nur für Sätze mit extrem wenigen, durchschnittlich ca. zehn Analysen angewandt. Die besten Ergebnisse liegen bei ca. 30% (Bod 2000), bzw. im Bereich von 55-75% (Johnson et al. (1999), Riezler et al. (2000)).

[16] Ein Parse wird hierbei in der gewöhnlichen Klammerschreibweise der Linguisten dargestellt.

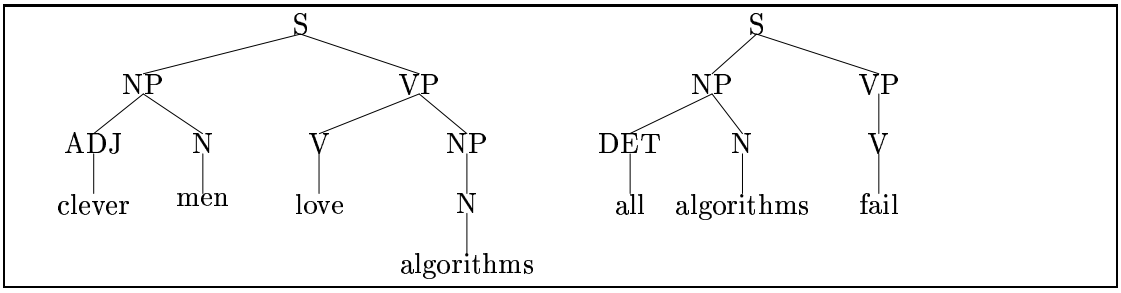


Abbildung 2.8: Beispiel-Baumbank.

$$
\begin{array}{rcl}
S & \rightarrow & NP\ VP \\
NP & \rightarrow & ADJ\ N \\
NP & \rightarrow & DET\ N \\
NP & \rightarrow & N \\
VP & \rightarrow & V\ NP \\
VP & \rightarrow & V \\
DET & \rightarrow & all \\
ADJ & \rightarrow & clever \\
N & \rightarrow & men \\
N & \rightarrow & algorithms \\
V & \rightarrow & love \\
V & \rightarrow & fail
\end{array}
$$

Abbildung 2.9: Aus der Spielzeug-Baumbank in Abbildung 2.8 abgelesene kontextfreie Grammatik.

Die *Sammlung* aller kontextfreien Regeln des Entwicklungskorpus dient in erster Linie dazu, eine kontextfreie Grammatik maschinell zu konstruieren und damit teure manuelle Entwicklungskosten einzusparen. Weitere Vorteile dieser Vorgehensweise sind, dass die Annotionskosten für die Herstellung der Baumbank teilweise kompensiert werden und die so abgeleitete Grammatik gut an die gegebene Baumbank angepasst ist. Der Sammelvorgang ist sehr einfach. Abbildung 2.8 zeigt eine Spielzeug-Baumbank mit zwei Syntaxbäumen. In Abbildung 2.9 werden alle kontextfreien Grammatikregeln gezeigt, die in einem der beiden Syntaxbäume vorkommen. Die aufgesammelten Regeln bilden eine kontextfreie Grammatik, die sogenannten *Baumbank-Grammatik*, mit welcher neue Sätze, zum Beispiel:

all men fail

abgeleitet werden können. Durch das Ablesen/Aufsammeln der Grammatikregeln von der Baumbank wurde also etwas gelernt. Das Aufsammeln der Grammatikregeln ist daher ein eigenständiges symbolisches Lernverfahren, das als Teil des im folgenden zu erläuternden stochastischen Lernverfahrens anzusehen ist. Der hier vorgestellte einfache Sammelvorgang wurde zuerst von Charniak (1996) auf ca. 300000 Wörter (grob geschätzt: 15000 Sätze) der Penn Wallstreet Journal Treebank angewandt, und ergab eine Baumbank-Grammatik mit ca. 10000 Grammatikregeln, von denen ca. 4000 Grammatikregeln mehr als einmal vorkamen[17].

Die *Auszählung* der Vorkommenshäufigkeiten der Grammatikregeln im Entwicklungskorpus ergibt eine *probabilistische kontextfreie Grammatik*. Der Schlüsselschritt ist überaus einfach: In Abschnitt 2.2 wurden probabilistische kontextfreie Grammatiken eingeführt. Die Standard-Nebenbedingung von Booth und Thompson (1973) für diese Grammatiken ist, dass die Grammatikregel-Wahrscheinlichkeiten auf allen Äquivalenzklassen G_A einer bestimmten Grammatikregelrelation eine Wahrscheinlichkeitsverteilung induzieren sollen:

$$\sum_{r \in G_A} p(r) = 1 \ .$$

Dabei gilt (siehe Abschnitt 2.2):

$$r \sim r' \quad :\Leftrightarrow \quad \mathrm{lhs}(r) = \mathrm{lhs}(r') \ ,$$

sowie (für geeignete Grammatikkategorien A):

$$G_A = \{r \in G \mid \mathrm{lhs}(r) = A\} \ .$$

[17] Müller (2001a) stellt eine sehr interessante Technik vor, in der die Regeln nicht nur aus einer gegebenen Baumbank aufgesammelt werden, sondern durch zusätzliche linguistische Informationen angereichert werden, mit dem Ziel, in der Anwendung die Disambiguierung alternativer Analysen zu unterstützen. Es wird experimentell gezeigt, dass die besten angereicherten Grammatiken gegenüber der "reinen Baumbank-Grammatik" eine verbesserte linguistische Performanz aufweisen und weniger Trainingsmaterial benötigen.

S	$\rightarrow$	$NP\ VP$	(1.0)
NP	$\rightarrow$	$ADJ\ N$	(0.33333)
NP	$\rightarrow$	$DET\ N$	(0.33333)
NP	$\rightarrow$	N	(0.33333)
VP	$\rightarrow$	$V\ NP$	(0.5)
VP	$\rightarrow$	V	(0.5)
DET	$\rightarrow$	*all*	(1.0)
ADJ	$\rightarrow$	*clever*	(1.0)
N	$\rightarrow$	*men*	(0.33333)
N	$\rightarrow$	*algorithms*	(0.66666)
V	$\rightarrow$	*love*	(0.5)
V	$\rightarrow$	*fail*	(0.5)

Abbildung 2.10: Aus der Spielzeug-Baumbank in Abbildung 2.8 abgelesene und trainierte probabilistische kontextfreie Grammatik.

In Abschnitt 2.2 wurde darauf hingewiesen, dass die Standard-Nebenbedingung weder notwendig noch hinreichend dafür ist, dass eine probabilistische kontextfreie Grammatik konsistent ist, d.h. eine Wahrscheinlichkeitsverteilung auf der Menge ihrer Syntaxbäume induziert. Befriedigend ist allerdings, dass Booth und Thompson (1973) zeigen konnten, dass eine zusätzliche Nebenbedingung die Konsistenz der Grammatik erzwingt. Ist $f(r)$ die Häufigkeit einer Grammatikregel r in der Baumbank, so kann die von der Baumbank abgelesene kontextfreie Grammatik mit der folgenden Definition zu einer probabilistischen kontextfreien Grammatik gemacht werden:

$$p(r) := \frac{f(r)}{\sum_{r \in G_A} f(r)}, \quad (r \in G_A) \ .$$

Natürlich folgt aus der Definition sehr einfach, dass diese stochastische Grammatik die Standard-Nebenbedingung für kontextfreie Grammatiken erfüllt. Dies ist eine sehr starke Begründung für die Wahl gerade dieser Wahrscheinlichkeitswerte. Bevor weitere gute Eigenschaften dieser Grammatik angeführt werden, soll die Beispiel-Grammatik aus Abbildung 2.9 "trainiert" werden. Da die Baumbank aus lediglich zwei Syntaxbäumen besteht, kann aus ihr wenig stochastische Information gezogen werden. Abbildung 2.10 zeigt, die mit dem Baumbank-Trainingsverfahren trainierte kontextfreie Grammatik. Interessant ist lediglich, dass die Grammatikregel $N \rightarrow$ *algorithms* in der Baumbank zweimal häufiger vorkommt, als die Grammatikregel $N \rightarrow$ *men*. Hieraus folgt zum Beispiel, dass der Satz

clever men fail (0.01388)

genau zweimal unwahrscheinlicher ist, als der Satz

$$\textit{clever algorithms fail} \qquad (0.02777) \ .$$

Die Satzwahrscheinlichkeiten sind in den runden Klammern angegeben und können leicht nachgerechnet werden. Bessere Grammatiken sollten berücksichtigen, wie oft Nomen als Subjekte oder als Objekte in einer Analyse vorkommen: Als Subjekt kommt *algorithms* in der Beispiel-Baumbank genauso oft vor wie *men*. Das sinnvolle Ergebnis dieser differenzierteren Vorgehensweise wäre, dass dann beide Beispielsätze gleich wahrscheinlich sind. Lexikalisierte Grammatiken berücksichtigen in der Regel die Subkategorisierungsrahmen von Verben und bewerten stochastisch, ob Nomen gute oder schlechte Argumente von Verben sind (*men love* versus *algorithms love*). Tatsächlich bringt diese verfeinerte Vorgehensweise ausgezeichnete stochastische Grammatiken hervor ((Carroll und Rooth 1998), (Beil et al. 1999)). Für echte Baumbanken (mit mehr als zwei Analysen) sind die stochastischen Implikationen natürlich wesentlich schlechter zu überblicken, als für die Beispiel-Baumbank. Umso interessanter ist, dass auch für grosse Baumbanken relativ einfach beweisbar ist, dass gerade Baumbank-Grammatiken unter den vielen möglichen probabilistischen Standard-Grammatiken stochastisch optimale Grammatiken sind. Dies wird im folgenden begründet. In der Definition der Grammatikregel-Wahrscheinlichkeiten für die Baumbank-Grammatik kommt auf der linken Seite eine empirische Wahrscheinlichkeit (siehe Abschnitt 2.1) vor:

$$\tilde{p}_A(r) := \frac{f(r)}{\sum_{r \in G_A} f(r)}, \quad (r \in G_A) \ .$$

Für diese empirische Wahrscheinlichkeitsverteilung $\tilde{p}_A(.)$ fungiert die Äquivalenzklasse G_A als Menge der Datentypen. Die von der Baumbank abgelesenen Regeln r mit einer linken Seite $\mathrm{lhs}(r) = A$, zusammen mit ihren Vorkommenshäufigkeiten $f(r)$ in der Baumbank, fungieren als Korpus:

$$f_A : \ G_A \to \mathcal{N}, \ r \mapsto f(r) \ .$$

Die trainierte Baumbank-Grammatik besteht also aus den empirischen Wahrscheinlichkeitsverteilungen $\tilde{p}_A(.)$ auf den Grammatikregel-Teilkorpora $f_A(.)$. In Abschnitt 2.4 wurde darauf hingewiesen, dass die empirische Wahrscheinlichkeitsverteilung die hervorragende Eigenschaft besitzt, die Korpuswahrscheinlichkeit zu maximieren. Übertragen auf die vorliegende Situation, bedeutet dies, dass die Baumbank-Grammatik die Wahrscheinlichkeiten aller Teilkorpora $f_A(.)$ maximiert:

$$\tilde{p}_A = \mathrm{argmax}_p \quad p(f_A) = \prod_{r \in G_A} p(r)^{f_A(r)} \ .$$

Dies ist ein schönes Ergebnis. Wünschenswert wäre aber auch, dass die trainierte Grammatik sogar der ganzen Baumbank (und nicht nur den Teilkorpora f_A) die grösstmögliche

Wahrscheinlichkeit zuweist. Auch dies ist der Fall. Der nun vorgestellte Beweis ist neu und greift auf die zuletzt hergeleiteten Ergebnisse zurück. Eine Baumbank ist ein Korpus $f_{\mathcal{T}}$ von Syntaxbäumen. Die in der Baumbank vorkommenden Syntaxbäume können als Syntaxbäume der von der Baumbank abgelesenen Baumbank-Grammatik G aufgefasst werden. Mit $\mathcal{T} := \mathcal{T}(G)$ gilt somit:

$$f_{\mathcal{T}} : \mathcal{T} \to \mathcal{N} .$$

Die trainierte Baumbank-Grammatik weist jedem Syntaxbaum $x \in \mathcal{T}$ eine Wahrscheinlichkeit $p(x)$ zu (siehe Abschnitt 2.2). Daher ist die Baumbank-Wahrscheinlichkeit eine gewöhnliche Korpuswahrscheinlichkeit (Abschnitt 2.1) und es gilt:

$$\begin{aligned}
p(f_{\mathcal{T}}) &:= \prod_{x \in \mathcal{T}} p(x)^{f_{\mathcal{T}}(x)} \\
&= \prod_{x \in \mathcal{T}} \left(\prod_{r \in G} p(r)^{f_r(x)} \right)^{f_{\mathcal{T}}(x)} \\
&= \prod_{x \in \mathcal{T}} \prod_{r \in G} p(r)^{f_r(x) \cdot f_{\mathcal{T}}(x)} \\
&= \prod_{r \in G} \prod_{x \in \mathcal{T}} p(r)^{f_r(x) \cdot f_{\mathcal{T}}(x)} \\
&= \prod_{r \in G} p(r)^{\sum_{x \in \mathcal{T}} f_r(x) \cdot f_{\mathcal{T}}(x)} \\
&= \prod_{r \in G} p(r)^{f(r)} \\
&= p(f) .
\end{aligned}$$

Die vorletzte Identität folgt, weil die Häufigkeit $f(r)$ einer Grammatikregel r in der Baumbank sich additiv aus ihren Vorkommenshäufigkeiten $f_r(x) \cdot f_{\mathcal{T}}(x)$ in allen Syntaxbäumen x der Baumbank zusammensetzt. Bei typweiser Durchsicht der Baumbank, muss die Häufigkeit $f_r(x)$ der Grammatikregel r im Syntaxbaum x natürlich mit der Häufigkeit $f_{\mathcal{T}}(x)$ des Syntaxbaums multipliziert werden:

$$f(r) = \sum_{x \in \mathcal{T}} f_r(x) \cdot f_{\mathcal{T}}(x) .$$

Die letzte Identität der Ableitung folgt mit der Definition der Korpuswahrscheinlichkeit. Interessant ist lediglich die damit verbundene Interpretation: Die Baumbank kann einerseits als Korpus von Syntaxbäumen aufgefasst werden, andererseits aber auch als Korpus von Grammatikregeln. Ein erstes schönes Ergebnis ist, dass die Wahrscheinlichkeit der Baumbank von der gewählten Interpretation unabhängig ist:

$$p(f_{\mathcal{T}}) = p(f) .$$

Dieses Ergebnis kann weiter verfeinert werden:

$$\begin{aligned} p(f_{\mathcal{T}}) &= \prod_{r \in G} p(r)^{f(r)} \\ &= \prod_{A} \prod_{r \in G_A} p(r)^{f(r)} \\ &= \prod_{A} \prod_{r \in G_A} p(r)^{f_A(r)} \\ &= \prod_{A} p(f_A) \ . \end{aligned}$$

Das sehr befriedigende Ergebnis ist also, dass die Baumbank-Wahrscheinlichkeit das Produkt der Wahrscheinlichkeiten aller Teilkorpora f_A ist. Natürlich kann dieses Ergebnis (nach dem oben erzielten Teilergebnis) nicht mehr überraschen: Da alle beteiligten Wahrscheinlichkeiten (Baumbank- und Teilkorpora-Wahrscheinlichkeiten) Produkte von Grammatikregel-Wahrscheinlichkeiten sind, und da bei der Aufteilung der Grammatikregeln in Teilkorpora f_A keine Grammatikregel doppelt gezählt wird, muss die soeben bewiesene Faktorisierung gelten. Alles zusammenfassend:

$$p(f_{\mathcal{T}}) = p(f) = \prod_{A} p(f_A) \ .$$

Da ein Produkt maximiert wird, indem alle seine Faktoren maximiert werden, gilt:

$$\begin{aligned} \text{argmax}_p \quad p(f_{\mathcal{T}}) &= \text{argmax}_p \quad \prod_{A} p(f_A) \\ &= \Big\langle \text{argmax}_p \quad p(f_A) \Big\rangle_A \\ &= \langle \tilde{p}_A \rangle_A \ . \end{aligned}$$

Der erste Schlüsselschritt dieses neuen Beweises eines bekannten Resultats ist, dass sich die Baumbank-Wahrscheinlichkeit als Produkt von Wahrscheinlichkeiten gewisser Teilkorpora darstellen lässt. Der zweite Schlüsselschritt ist, dass die empirische Wahrscheinlichkeitsverteilung $\tilde{p}_A(.)$ die Korpuswahrscheinlichkeit $p(f_A)$ unter allen Wahrscheinlichkeitsverteilungen $p(.)$ maximiert. Aus beiden Schritten ergibt sich dann leicht, dass die Baumbank-Wahrscheinlichkeit mit der Baumbank-Grammatik maximiert wird, solange für die Maximierung nur Grammatiken mit Standard-Nebenbedingung (Booth und Thompson 1973) betrachtet werden[18]. Dieses schöne Ergebnis soll diesen Abschnitt beschliessen.

[18] **Bemerkung**: Möglicherweise gibt es probabilistische kontextfreie Grammatiken, die der Baumbank eine höhere Wahrscheinlichkeit zuweisen, als die Baumbank-Grammatik. Offenbar verletzen solche Grammatiken die Standard-Nebenbedingung für probabilistische kontextfreie Grammatiken, welche aber für die Konsistenz von Grammatiken weder notwendig noch hinreichend ist (Booth und Thompson 1973). Daher gilt: ***Möglicherweise gibt es konsistente Grammatiken mit einer höheren Baumbank-Wahrscheinlichkeit, als die Baumbank-Grammatik.*** Es könnte daher eine lohnende Aufgabe sein, nach

2.6 Stochastische Parsingverfahren

In diesem Abschnitt wird die Benutzung einer trainierten stochastischen kontextfreien Grammatik vorgestellt. Natürlich können stochastische kontextfreie Grammatiken in allen Fällen eingesetzt werden, in denen nicht-stochastische kontextfreie Grammatiken eingesetzt werden. Umgekehrt ist dies leider nicht der Fall, da die stochastischen kontextfreien Grammatiken aufgrund ihrer Regelwahrscheinlichkeiten wesentlich vielfältiger eingesetzt werden können. Um diese Unterscheidung möglichst klar auszudrücken, soll im folgenden von symbolischen und stochastischen Parsingverfahren gesprochen werden:

Die Vorgehensweise, bei der für einen Satz alle seine möglichen Analysen (für kontextfreie Grammatiken seine Syntaxbäume) berechnet werden, wird *symbolisches Parsingverfahren* genannt. Im Vergleich hierzu ist ein *stochastisches Parsingverfahren* ein Vorgang, bei dem für einen Satz nicht nur seine Analysen, sondern auch deren Wahrscheinlichkeiten (oder andere relevante Zählgrössen) berechnet werden.

Für symbolisches Parsing mit kontextfreien Grammatiken sind seit vielen Jahren wohl ebenso viele Parsing-Algorithmen bekannt, von denen der sogenannte CKY-Algorithmus vielleicht der bekannteste ist. Auch für stochastisches Parsing gibt es eine Vielzahl von Parsing-Algorithmen. Während sich aber die symbolischen Parsing-Algorithmen vor allem darin unterscheiden, welche Grammatik-Voraussetzungen sie fordern und auf welche spezielle Art und Weise sie die Satzanalysen herstellen und verwalten, ist die Unterscheidung der stochastischen Parsing-Algorithmen von prinzipiellerer Art.

Hierbei geht es primär um die Frage, was ein stochastischer Parsing-Algorithmus, neben der Herstellung der Satzanalysen, zusätzlich leisten soll. Die folgende Aufstellung gibt hierzu Auskunft, in die auch der Count-Algorithmus aufgenommen wurde, weil dieser, auf rein symbolische kontextfreie Grammatiken anwendbare Algorithmus, seiner ganzen Natur nach, eher den stochastischen Parsingverfahren zuzuordnen ist:

- *Count-Algorithmus*: Berechnung der Anzahl der Satzanalysen (bzw. der Teilsatz--Analysen),

- *Inside-Algorithmus*: Berechnung der *Inside-Wahrscheinlichkeit*, d.h. der Satzwahrscheinlichkeit (bzw. der Teilsatz-Wahrscheinlichkeiten),

konsistenten Non-Standard-Modellen zu suchen, welche bessere stochastische und damit vermutlich auch bessere linguistische Eigenschaften, als die Baumbank-Grammatik, aufweisen. Chi beweist (Chi (1998), Chi (1999)), dass Baumbank-Grammatiken und die mit dem Inside-Outside-Algorithmus estimierten Grammatiken konsistent sind, wenn keine zusätzlichen Techniken, wie Smoothing von Grammatikregeln, etc. eingesetzt werden.

- *Viterbi-Algorithmus*: Berechnung der *Viterbi-Wahrscheinlichkeit*, d.h. der maximalen Wahrscheinlichkeit der Satzanalysen (bzw. der Teilsatz-Analysen),

Eine besonders vorzügliche Eigenschaft probabilistischer kontextfreier Grammatiken ist es, dass diese, dank des Viterbi-Algorithmus (bzw. des Viterbi-Parse-Algorithmus, siehe Abschnitt 2.7), effizient zur Disambiguierung herangezogen werden können, auch wenn, in einer realistischen Grammatik, ein Satz in der Regel mehrere zehntausend Analysen besitzt (Rooth et al. 1999).

Diese Eigenschaft hebt probabilistische kontextfreie Grammatiken einerseits aus der Menge der rein symbolischen Grammatiken heraus, mit denen prinzipiell keine Disambiguierung möglich ist, wenn ein Satz mehr als eine einzige Satzanalyse aufweist.

Wichtiger ist jedoch, dass der Viterbi-Algorithmus kontextfreie Grammatiken in die Lage versetzt, *effizient* zu disambiguieren, was diese auch innerhalb der nicht eben zahlreichen Menge aller bekannten stochastischen Grammatiken, insbesondere der stochastischen unifikationsbasierten Grammatiken, äusserst positiv heraushebt, da nur diese Eigenschaft eine praktische Anwendung einer stochastischen Grammatik bewirken kann.

Bis zum heutigen Zeitpunkt ist für stochastische unifikationsbasierte Grammatiken kein effizienter Algorithmus zur Bestimmung des wahrscheinlichsten Parses bekannt. Dies hat zur Folge, dass stochastische unifikationsbasierte Grammatiken leider nur in sehr eingeschränktem Umfang zur Disambiguierung beliebiger Satzanalysen zu gebrauchen sind; wobei sie allerdings für Sätze mit wenigen dutzend Analysen, für die sie anwendbar sind, gute Ergebnisse aufweisen (Riezler et al. 2000).

Für das weitere Vorgehen ist es vorteilhaft, sich den Algorithmus von Cocke, Kasami und Younger (Kasami 1965) in Erinnerung zu rufen. Der CKY-Algorithmus löst für eine gegebene kontext-freie Grammatik in Chomsky-Normalform das Parsingproblem, d.h. die Fragestellung, ob ein gegebener Satz grammatisch ist, oder in anderen Worten, ob eine gegebene Wortfolge w die Satzkategorie S tragen kann. Da der CKY-Algorithmus auf dem Prinzip der dynamischen Programmierung basiert, ist er ein sehr effizienter Algorithmus. *Dynamische Programmierung* wird dabei jeder Algorithmus genannt, der ein Problem löst, indem er das gegebene Problem in kleinere Unterprobleme zerlegt und dann die Lösung rekursiv so berechnet, *dass jedes Teilproblem nur einmal gelöst wird.*[19]

Beim CKY-Algorithmus sieht das konkret so aus, dass er die etwas allgemeinere Fragestellung, ob ein beliebiger Teilsatz w_A (eine beliebige Teil-Wortfolge des Satzes w) eine bestimmte grammatikalische Kategorie A (zum Beispiel NP, DET, N, VP, V, ...) tragen

[19]Der angegebene Zusatz ist sehr wichtig. Im Parsing macht dies den Unterschied zwischen Chartparsern und anderen Parsern mir exponentieller Laufzeit (zum Beispiel Recursive-Decent-Parsern) aus.

kann, rekursiv zu beantworten sucht. Bezeichnet $\mathcal{T}(G)$ die Menge aller Syntaxbäume einer kontextfreien Grammatik G, so ist der Schlüsselschritt, dass gezeigt wird, dass

$$A \Rightarrow^* w_A := \{x_A \in \mathcal{T}(G) \mid x_A \text{ hat die Wurzel } A \text{ und die Blattfolge } w_A\}$$

unter milden Grammatik-Voraussetzungen rekursiv aus irgendwelchen Mengen $B \Rightarrow^* w_B$ mit $|w_B| < |w_A|$ berechnet werden kann.

Als Voraussetzung des CKY-Algorithmus wird angenommen, dass die kontextfreie Grammatik in *Chomsky-Normalform* vorliegt[20], d.h. dass alle Grammatikregeln entweder von der Form

$$A \to B\ C \quad \text{oder} \quad A \to a$$

sind; hierbei ist a ein beliebiges Wort aus dem Grammatiklexikon und A, B, C sind beliebige Grammatikkategorien. Unter dieser milden Voraussetzung, die insbesondere die wenig eleganten Epsilon- und Kettenregeln ausschliessen, kann es offenbar für ein einzelnes Wort $w_A = a$ nur einen einzigen Syntaxbaum mit Wurzelknoten A und Blattknoten a geben:

$$A \Rightarrow^* w_A \ = \ \{A \to a\}\ .$$

Für eine Wortfolge w_A mit $|w_A| \geq 2$ gilt jedoch:

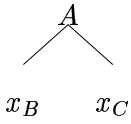
$$A \Rightarrow^* w_A \ = \ \left\{ \begin{array}{c} A \\ x_B \quad x_C \end{array} \ \middle| \ \begin{array}{c} A \to BC, \\ w_A = w_B w_C, \\ x_B \in\ B \Rightarrow^* w_B, \\ x_C \in\ C \Rightarrow^* w_C \end{array} \right\}$$

$$= \sum_{\substack{A \to BC, \\ w_A = w_B w_C}} \left\{ \begin{array}{c} A \\ x_B \quad x_C \end{array} \ \middle| \ \begin{array}{c} x_B \in\ B \Rightarrow^* w_B, \\ x_C \in\ C \Rightarrow^* w_C \end{array} \right\}$$

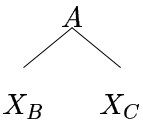
$$= \sum_{\substack{A \to BC, \\ w_A = w_B w_C}} \begin{array}{c} A \\ B \Rightarrow^* w_B \quad C \Rightarrow^* w_C \end{array} \ .$$

[20] Für theoretische Überlegungen, nicht aber in der Praxis, ist die Annahme von Chomsky-Normalform eine milde Anforderung, da jede kontextfreie Grammatik in eine Grammatik in Chomsky-Normalform umgewandelt werden kann. Ihre Grösse kann dabei aber exponentiell anwachsen und - was schlimmer ist - es ist schwer, aus den Analysen der Grammatik in Chomsky-Normalform die Analysen der ursprünglichen Grammatik zu rekonstruieren. Wegen der grösseren praktischen Bedeutung und der Anwendbarkeit der hier vorgestellten Parsewald-Algorithmen auf allgemeine kontextfreie Grammatiken, wäre es natürlich sehr interessant, die Parseald-Algorithmen auch für den Earley-Algorithmus zu formulieren.

Hierbei weist die Notation $\sum$ daraufhin, dass es sich um eine Vereinigung disjunkter Mengen handelt. Ferner wurde für einen Syntaxbaum x_B mit Wurzel B und einen Syntaxbaum x_C mit Wurzel C und für eine Regel $A \to BC$ das Symbol



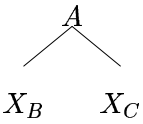
für den aus diesen drei Teilen in natürlicher Weise zusammengesetzten Syntaxbaum mit der Wurzel A benutzt. Schliesslich benutzt man für eine Menge X_A von Syntaxbäumen mit Wurzel A und eine Menge X_B von Syntaxbäumen mit Wurzel B und für eine Regel $A \to BC$ naheliegenderweise das Symbol



für die folgende Menge von Syntaxbäumen:

$$\left\{ \begin{array}{c} A \\ x_B \quad x_C \end{array} \;\middle|\; x_B \in X_B,\ x_C \in X_C \right\} .$$

Diese Definition erinnert sehr an die Definition des *Kreuzprodukts* in der Mengentheorie. Wie dort, ergibt



die leere Menge, wenn die Menge X_B oder die Menge X_C leer ist.

Die Formel:

$$A \Rightarrow^* w_A \;=\; \begin{cases} \{A \to w_A\} & \text{für } |w_A| = 1 \\ \displaystyle\sum_{\substack{A \to BC, \\ w_A = w_B w_C}} \begin{array}{c} A \\ B \Rightarrow^* w_B \quad C \Rightarrow^* w_C \end{array} & \text{für } |w_A| \geq 2 \end{cases}$$

beleuchtet in sehr anschaulicher Weise den rekursiven Charakter des Parsingproblems, d.h. der Aufgabe der Berechnung von $A \Rightarrow^* w_A$.

Während die erste Alternative einen Induktionsanfang begründen kann, ist es möglich, die zweite Alternative für einen Induktionsschluss zu benutzen, weil die Berechnung der Menge

Eingabe: Eine Grammatik in Chomsky-Normalform und ein Satz $w = w_1 \dots w_n$.

Ausgabe: Die Parsewälder $forest[s, A, s+l]$ für alle Teilsätze der Länge $1 \leq l \leq n$, Startindizes $1 \leq s \leq n+1-l$, und Wurzelkategorien A.

Prozedur:

1. $forest[.,.,.] := \emptyset$;
2. **for each** $s := 1, \dots, n$ **do**
3. **for each** $A \to w_s$ **do**
4. $forest[s, A, s+1] := \{A \to w_s\}$;
5. **for each** $l := 2, \dots, n$ **do**
6. **for each** $s := 1, \dots, n+1-l$ **do**
7. **for each** $t := 1, \dots, l-1$ **do**
8. **for each** $A \to B\ C$ **do**
9. $forest[s, A, s+l] := forest[s, A, s+l] +$
10. 

Abbildung 2.11: Parsewald-Algorithmus

$A \Rightarrow^* w_A$ auf die Berechnung der Mengen $B \Rightarrow^* w_B$ und $C \Rightarrow^* w_C$ zurückgeführt wurde, und diese sich wegen $|w_B| < |w_A|$ und $|w_C| < |w_A|$ als Teilprobleme des Ausgangsproblems auffassen lassen.

Abbildung 2.11 zeigt einen Algorithmus, der treffenderweise *Parsewald-Algorithmus* genannt werden könnte, weil er die Syntaxbäume sämtlicher Teilsätze dieser Induktionsidee folgend, verwaltet.

Die Eingabe des Parsewald-Algorithmus besteht aus einer Grammatik in Chomsky-Normalform und einem Satz $w = w_1 \dots w_n$, $(n \geq 1)$. Die Ausgabe des Parsewald-Algorithmus besteht aus den Parsewäldern

$$forest[s, A, s+l] \ := \ A \Rightarrow^* w_s \dots w_{s+l-1}$$

d.h. den Syntaxbäumen aller Teilsätze der *Länge* $1 \leq l \leq n$ und *Startindex* $1 \leq s \leq n+1-l$, mit der Grammatikkategorie A als Topknoten.

Man versteht den Algorithmus, insbesondere die 9.-10. Zeile problemlos, wenn man sich klar macht, dass in der Prozedur im Vergleich zu den vorherigen Ausführungen die folgende

Eingabe: Eine Grammatik in Chomsky-Normalform und ein Satz $w = w_1 \dots w_n$.

Ausgabe: Die Parsechart $chart[s, A, s+l] \in \{TRUE, FALSE\}$ für alle Teilsätze der Länge $1 \leq l \leq n$, Startindizes $1 \leq s \leq n+1-l$, und Grammatikkategorien A.

Prozedur:

```
1.  chart[.,.,.] := FALSE;
2.  for each s := 1,...,n do
3.      for each A → w_s do
4.          chart[s,A,s+1] := TRUE;
5.  for each l := 2,...,n do
6.      for each s := 1,...,n+1-l do
7.          for each t := 1,...,l-1 do
8.              for each A → B C do
9.                  chart[s,A,s+l] := chart[s,A,s+l] ∨
10.                     chart[s,B,s+t] ∧ chart[s+t,C,s+l]
```

Abbildung 2.12: CKY-Algorithmus

Notation benutzt wird:

$$w_A = w_s \dots w_{s+l-1}, \quad w_B = w_s \dots w_{s+t-1} \quad \text{und} \quad w_C = w_{s+t} \dots w_{s+l-1},$$

sodass, wie vorher ausgeführt:

$$w_A = w_B w_C$$

gilt. Das Füllen der Parsewälder geschieht induktiv für monoton grösser werdende Teilsätze w_A der Längen $l = 1, 2, \dots, n$. In der 1. Zeile der Prozedur werden die Parsewälder als leere Mengen initialisiert, während in der 2.-4. Zeile der Induktionsanfang und in der 5.-10. Zeile der Induktionsschluss stattfindet. Interessant ist das Zeichen $'+'$, welches in der 9. Zeile anstatt des Mengenvereinigungszeichens $'\cup'$ benutzt wird und anzeigen soll, dass eine Vereinigung disjunkter Mengen stattfindet. Da die Grammatik fest ist, können die 3. und 4. Zeile, sowie die 8.-10. Zeile in konstanter Zeit abgearbeitet werden, wobei man sich lediglich Gedanken um eine angemessene Implementierung der Parsewälder machen muss. Die Abarbeitung der 2.-4. bzw. der 5.-10. Zeile benötigt daher $O(n)$ bzw. $O(n^3)$ Zeit. Der Parsewald-Algorithmus ist somit von der Ordnung $O(n^3)$.

In Abbildung 2.12 ist der CKY-Algorithmus wiedergegeben. Die Eingabe des CKY-Algorithmus besteht aus einer Grammatik in Chomsky-Normalform und einem Satz $w = w_1 \dots w_n$, $(n \geq 1)$, dessen Grammatikalität zu überprüfen ist. Die Ausgabe des

CKY-Algorithmus besteht aus der sogenannten *Parsechart* $chart[.,.,.]$, die für einen Teilsatz der *Länge* $1 \leq l \leq n$ und dessen *Startindex* $1 \leq s \leq n+1-l$, sowie für eine Grammatikkategorie A einen Boole'schen Wert

$$chart[s, A, s+l] \in \{TRUE, FALSE\}$$

wie folgt annimmt:

$$chart[s, A, s+l] = \begin{cases} TRUE & \text{falls } A \Rightarrow^* w_s \ldots w_{s+l-1} \text{ nicht leer ist ,} \\ FALSE & \text{sonst .} \end{cases}$$

Der CKY-Algorihmus verwaltet also, anstatt der Parsewälder $forest[s, A, s+l]$, lediglich die Informationen $chart[s, A, s+l]$, unabhängig ob die Parsewälder leer sind oder nicht. Der Chart-Algorithmus ist daher eine einfache Modifikation des Parsewald-Algorithmus, die im wesentlichen auf den folgenden Booleschen Gleichungen beruht (für $|w_A| \geq 2$):

$$chart(\ A \Rightarrow^* w_A\ ) := \begin{cases} TRUE & \text{Die folgende Menge ist nicht leer:} \\ & \bigvee_{\substack{A \to BC, \\ w_A = w_B w_C}} \quad \begin{array}{c} A \\ B \Rightarrow^* w_B \qquad C \Rightarrow^* w_C \end{array} \\ FALSE & \text{sonst} \end{cases}$$

$$= \bigvee_{\substack{A \to BC, \\ w_A = w_B w_C}} \begin{cases} TRUE & \text{Die folgende Menge ist nicht leer:} \\ & \begin{array}{c} A \\ B \Rightarrow^* w_B \qquad C \Rightarrow^* w_C \end{array} \\ FALSE & \text{sonst} \end{cases}$$

$$= \bigvee_{\substack{A \to BC, \\ w_A = w_B w_C}} \left(chart(B \Rightarrow^* w_B) \wedge\ chart(C \Rightarrow^* w_C) \right) .$$

Da der CKY-Algorithmus eine Chart benutzt und das Füllen der Chart für Wortfolgen mit der kleinster Länge 1 beginnt und für monoton grösser werdende Wortfolgen mit der Wortfolge von maximaler Länge n endet, wird der CKY-Parser auch ein *Bottom-Up-Chart-Parser* genannt. Der CKY-Algorithmus ist wie der Parsewald-Algorithmus von der Ordnung $O(n^3)$.

Der sogenannte *Count-Algorithmus* ist der erste Parsing-Algorithmus, der als ein stochastischer Parsing-Algorithmus vorgestellt werden soll. Er ist in Abbildung 2.13 wiedergegeben.

Eingabe: Eine Grammatik in Chomsky-Normalform und ein Satz $w = w_1 \dots w_n$.

Ausgabe: Die Anzahl $count[s, A, s+l]$ der grammatischen Analysen für alle Teilsätze der Länge $1 \leq l \leq n$, Startindizes $1 \leq s \leq n+1-l$, und Grammatikkategorien A.

Prozedur:

1. $count[.,.,.] := 0$;
2. **for each** $s := 1, \dots, n$ **do**
3. **for each** $A \to w_s$ **do**
4. $count[s, A, s+1] := 1$;
5. **for each** $l := 2, \dots, n$ **do**
6. **for each** $s := 1, \dots, n+1-l$ **do**
7. **for each** $t := 1, \dots, l-1$ **do**
8. **for each** $A \to B\ C$ **do**
9. $count[s, A, s+l] := count[s, A, s+l] +$
10. $count[s, B, s+t] \cdot count[s+t, C, s+l]$

Abbildung 2.13: Count-Algorithmus

Obwohl seine Eingabe wie beim Parsewald-Algorithmus und dem CKY-Algorithmus nur aus einer rein symbolischen Grammatik besteht, ist der Count-Algorithmus seiner Natur nach eher ein stochastischer Algorithmus, da er Häufigkeiten (die Zahl der Analysen aller Teilsätze), und damit stochastische Basisgrössen, verwaltet. Ein weiterer Grund, weshalb der Count-Algorithmus eher zu den stochastischen Parsing-Algorithmen gezählt werden sollte, ist seine grosse Ähnlichkeit mit dem Inside-Algorithmus, welcher statt Häufigkeiten Wahrscheinlichkeiten von Analysen berechnet.

Die Eingabe des Count-Algorithmus besteht aus einer Grammatik in Chomsky-Normalform und einem Satz $w = w_1 \dots w_n,\ (n \geq 1)$, dessen Anzahl der Syntaxbäume zu ermitteln ist. Die Ausgabe des Count-Algorithmus besteht aus den sogenannten *Counts*:

$$count[s, A, s+l] \ := \ \mid A \Rightarrow^* w_s \dots w_{s+l-1} \mid .$$

die für einen Teilsatz der *Länge* $1 \leq l \leq n$ und dessen *Startindex* $1 \leq s \leq n+1-l$ die Anzahl seiner Syntaxbäume mit Topknoten A angibt.

Auch der Count-Algorithmus ist nur eine einfache Modifikation des Parsewald-Algorithmus, weil für eine Wortfolge w_A mit $|w_A| \geq 2$ die folgende Identität gilt:

$$\begin{aligned}
count(A \Rightarrow^* w_A) \ &:= \ \mid A \Rightarrow^* w_A \mid \\
&= \left| \sum_{\substack{A \to BC, \\ w_A = w_B w_C}} \quad \begin{array}{ccc} & A & \\ \diagup & & \diagdown \\ B \Rightarrow^* w_B & & C \Rightarrow^* w_C \end{array} \right| \\
&= \left| \sum_{\substack{A \to BC, \\ w_A = w_B w_C}} \quad \begin{array}{ccc} & A & \\ \diagup & & \diagdown \\ B \Rightarrow^* w_B & & C \Rightarrow^* w_C \end{array} \right| \\
&= \sum_{\substack{A \to BC, \\ w_A = w_B w_C}} |B \Rightarrow^* w_B| \cdot |C \Rightarrow^* w_C| \\
&= \sum_{\substack{A \to BC, \\ w_A = w_B w_C}} count(B \Rightarrow^* w_B) \cdot count(C \Rightarrow^* w_C) \ .
\end{aligned}$$

In der 1. Zeile des Count-Algorithmus werden die Analyseanzahlen mit 0 initialisiert, während in der 2.-4. Zeile der Induktionsanfang und in der 5.-10. Zeile stattfindet. Die

Eingabe: Eine probabilistische Grammatik in Chomsky-Normalform und ein Satz $w = w_1 \dots w_n$.

Ausgabe: Die Inside-Wahrscheinlichkeiten $inside[s, A, s+l] \in [0, 1]$ für alle Teilsätze der Länge $1 \leq l \leq n$, Startindizes $1 \leq s \leq n+1-l$, und Grammatikkategorien A.

Prozedur:

1. $inside[.,.,.] := 0$;
2. **for each** $s := 1, \dots, n$ **do**
3. **for each** $A \to w_s$ **do**
4. $inside[s, A, s+1] := p(A \to w_s)$;
5. **for each** $l := 2, \dots, n$ **do**
6. **for each** $s := 1, \dots, n+1-l$ **do**
7. **for each** $t := 1, \dots, l-1$ **do**
8. **for each** $A \to B\ C$ **do**
9. $inside[s, A, s+l] := inside[s, A, s+l] +$
10. $p(A \to B\ C) \cdot inside[s, B, s+t] \cdot inside[s+t, C, s+l]$;

Abbildung 2.14: Inside-Algorithmus

eben hergeleitete Identität wird in der 9. Zeile angewandt. Selbstverständlich ist auch der Count-Algorithmus von der Ordnung $O(n^3)$.

Der sogenannte *Inside-Algorithmus*, der in Abbildung 2.14 wiedergegeben ist, ist der erste, der hier vorgestellten Parsing-Algorithmen, der tatsächlich eine nicht-symbolische Grammatik als Eingabe fordert und somit als erster "echter stochastischer Parsing-Algorithmus" gelten kann:

Die Eingabe des Inside-Algorithmus besteht aus einer probabilistischen Grammatik in Chomsky-Normalform und einem Satz $w = w_1 \dots w_n$, $(n \geq 1)$, dessen Analyse-Wahrscheinlichkeit zu ermitteln ist. Die Ausgabe des Inside-Algorithmus besteht aus den sogenannten *Inside-Wahrscheinlichkeiten*:

$$inside[s, A, s+l] \ := \ p(A \Rightarrow^* w_s \dots w_{s+l-1})$$

für alle Teilsätze der *Länge* $1 \leq l \leq n$ und *Startindizes* $1 \leq s \leq n+1-l$, sowie Grammatikkategorien A. Die Inside-Wahrscheinlichkeit eines Teilsatzes ist somit die Summe der Wahrscheinlichkeiten aller Syntaxbäume (mit Wurzelkategorie A) dieses Teilsatzes und unterscheidet sich von dessen Count nur dadurch, dass ein Syntaxbaum mit seiner

Wahrscheinlichkeit, anstatt der Zahl 1, gezählt wird. Auch der Inside-Algorithmus ist eine einfache Modifikation des Parsewald-Algorithmus, weil für eine Wortfolge w_A mit $|w_A| \geq 2$ die folgenden Umformungen möglich sind, und diese ein ähnliches Ergebnis wie beim Count-Algorithmus liefern:

$$
\begin{aligned}
\mathit{inside}(\ A \Rightarrow^* w_A) \ &:= \ p(\ A \Rightarrow^* w_A) \\
&= \ p\left(\sum_{\substack{A \to BC, \\ w_A = w_B w_C}} \ \begin{array}{c} A \\ \diagup \quad \diagdown \\ B \Rightarrow^* w_B \qquad C \Rightarrow^* w_C \end{array} \right) \\
&= \ \sum_{\substack{A \to BC, \\ w_A = w_B w_C}} p\left(\begin{array}{c} A \\ \diagup \quad \diagdown \\ B \Rightarrow^* w_B \qquad C \Rightarrow^* w_C \end{array} \right) \\
&= \ \sum_{\substack{A \to BC, \\ w_A = w_B w_C}} \ \sum_{\substack{x_B \in B \Rightarrow^* w_B \\ x_C \in C \Rightarrow^* w_C}} p\left(\begin{array}{c} A \\ \diagup \quad \diagdown \\ x_B \qquad x_C \end{array} \right) \\
&= \ \sum_{\substack{A \to BC, \\ w_A = w_B w_C}} \ \sum_{\substack{x_B \in B \Rightarrow^* w_B \\ x_C \in C \Rightarrow^* w_C}} p(A \to BC) \cdot p(x_B) \cdot p(x_C) \\
&= \ \sum_{\substack{A \to BC, \\ w_A = w_B w_C}} p(A \to B\ C) \cdot \sum_{\substack{x_B \in B \Rightarrow^* w_B \\ x_C \in C \Rightarrow^* w_C}} p(x_B) \cdot p(x_C) \\
&= \ \sum_{\substack{A \to BC, \\ w_A = w_B w_C}} p(A \to B\ C) \sum_{x_B \in B \Rightarrow^* w_B} p(x_B) \sum_{x_C \in C \Rightarrow^* w_C} p(x_C) \\
&= \ \sum_{\substack{A \to BC, \\ w_A = w_B w_C}} p(A \to B\ C) \cdot p(B \Rightarrow^* w_B) \cdot p(C \Rightarrow^* w_C)
\end{aligned}
$$

Eingabe: Eine probabilistische Grammatik in Chomsky-Normalform und ein Satz $w = w_1 \dots w_n$.

Ausgabe: Die Viterbi-Wahrscheinlichkeiten $Viterbi[s, A, s+l] \in [0,1]$ für alle Teilsätze der Länge $1 \leq l \leq n$, Startindizes $1 \leq s \leq n+1-l$, und Grammatikkategorien A.

Prozedur:

1. $Viterbi[.,.,.] := 0$;
2. **for each** $s := 1, \dots, n$ **do**
3. **for each** $A \to w_s$ **do**
4. $Viterbi[s, A, s+1] := p(A \to w_s)$;
5. **for each** $l := 2, \dots, n$ **do**
6. **for each** $s := 1, \dots, n+1-l$ **do**
7. **for each** $t := 1, \dots, l-1$ **do**
8. **for each** $A \to B\ C$ **do**
9. $Viterbi[s, A, s+l] := \max\{Viterbi[s, A, s+l],$
10. $p(A \to B\ C) \cdot Viterbi[s, B, s+t] \cdot Viterbi[s+t, C, s+l]\}$;

Abbildung 2.15: Viterbi-Algorithmus

$$= \sum_{\substack{A \to BC, \\ w_A = w_B w_C}} p(A \to B\ C) \cdot inside(B \Rightarrow^* w_B) \cdot inside(C \Rightarrow^* w_C)\ .$$

In der 1. Zeile des Inside-Algorithmus werden die Inside-Wahrscheinlichkeiten mit 0 initialisiert, während in der 2.-4. Zeile der Induktionsanfang und in der 5.-10. Zeile stattfindet. Die eben hergeleitete Identität wird in der 9. Zeile angewandt. Auch der Inside-Algorithmus ist von der Ordnung $O(n^3)$.

Der sogenannte *Viterbi-Algorithmus*, der in Abbildung 2.15 wiedergegeben ist, ist vermutlich der bekannteste aller stochastischen Parsing-Algorithmen, da er das Disambiguierungsproblem angeht, indem er die grösste Wahrscheinlichkeit aller Satzanalysen ermittelt.

Die Eingabe des Viterbi-Algorithmus besteht aus einer probabilistischen Grammatik in Chomsky-Normalform und einem Satz $w = w_1 \dots w_n$, $(n \geq 1)$, dessen grösste Analyse-Wahrscheinlichkeit zu ermitteln ist. Die Ausgabe des Viterbi-Algorithmus besteht aus den sogenannten *Viterbi-Wahrscheinlichkeiten*:

$$Viterbi[s, A, s+l] \ := \ \max\{p(x_A) \mid x_A \in A \Rightarrow^* w_s \dots w_{s+l-1}\}$$

für alle Teilsätze der *Länge* $1 \leq l \leq n$ und *Startindizes* $1 \leq s \leq n+1-l$, sowie Grammatikkategorien A. Die Viterbi-Wahrscheinlichkeit eines Teilsatzes ist somit die grösste Wahrscheinlichkeit aller Syntaxbäume (mit Wurzelkategorie A) dieses Teilsatzes. Wie alle bisherigen Algorithmen ist auch der Viterbi-Algorithmus eine einfache Modifikation des Parsewald-Algorithmus, was darauf zurückzuführen ist, dass für eine Wortfolge w_A mit $|w_A| \geq 2$ die folgende Identität gilt:

$$
\begin{aligned}
Viterbi(A \Rightarrow^* x_A) &:= \max\{p(x_A) \mid x_A \in A \Rightarrow^* x_A\} \\
&= \max\left\{ p\left(\begin{array}{c} A \\ \diagup\ \diagdown \\ x_B \quad x_C \end{array} \right) \;\middle|\; \begin{array}{c} A \to BC, \\ w_A = w_B w_C, \\ x_B \in B \Rightarrow^* w_B, \\ x_C \in C \Rightarrow^* w_C \end{array} \right\} \\
&= \max\left\{ p(A \to BC) \cdot p(x_B) \cdot p(x_C) \;\middle|\; \begin{array}{c} A \to BC, \\ w_A = w_B w_C, \\ x_B \in B \Rightarrow^* w_B, \\ x_C \in C \Rightarrow^* w_C \end{array} \right\} \\
&= \max\left\{ \begin{array}{c} p(A \to BC) \cdot \\ \max\{p(x_B) | x_B \in B \Rightarrow^* w_B\} \cdot \\ \max\{p(x_C) | x_C \in C \Rightarrow^* w_C\} \end{array} \;\middle|\; \begin{array}{c} A \to BC, \\ w_A = w_B w_C \end{array} \right\} \\
&= \max\left\{ \begin{array}{c} p(A \to BC) \cdot \\ Viterbi(B \Rightarrow^* x_B) \cdot \\ Viterbi(C \Rightarrow^* x_C) \end{array} \;\middle|\; \begin{array}{c} A \to BC, \\ w_A = w_B w_C \end{array} \right\} .
\end{aligned}
$$

In der 1. Zeile des Viterbi-Algorithmus in Abbildung 2.15 werden die Inside-Wahrscheinlichkeiten mit 0 initialisiert, während in der 2.-4. Zeile der Induktionsanfang und in der 5.-10. Zeile stattfindet. Die eben hergeleitete Identität wird in der 9. Zeile angewandt. Der Viterbi-Algorithmus ist von der Ordnung $O(n^3)$.

Damit soll dieser Abschnitt beschlossen werden. Drei weitere noch nicht besprochene stochastische Parsingverfahren, der Viterbi-Parse-, der N-Best- und der N-Best-Parse-Algorithmus, sind dem Viterbi-Algorithmus sehr ähnlich. Sie werden im Abschnitt 2.7 über State-of-the-Art Parsingverfahren als Instanzen des Semiring-Parsing-Algorithmus eingeführt.

2.7 State-of-the-Art Parsingverfahren

Die Überschrift deutet darauf hin, dass in diesem Abschnitt die allerneuesten symbolischen und stochastischen Parsingverfahren vorgestellt werden. Hierzu wird jedoch auf das Kapitel 5 verwiesen, in welchem Experimente zu lexikalisierten probabilistischen kontextfreien Grammatiken und stochastischen Unifikationsgrammatiken vorgestellt werden. Dieser Abschnitt ist vielmehr einer Arbeit gewidmet, die zu einer völlig neuen Auffassung der meisten effizienten, dynamischen Parsingverfahren für kontextfreie Grammatiken führte.

Obwohl, vielleicht auch weil, der CKY-Algorithmus (Kasami 1965) schon relativ alt ist, wurde mit seiner Hilfe eine aufregende Entdeckung für stochastische Grammatiken gemacht: Goodman (1998) wies in jüngster Zeit nach, dass eine Vielzahl der symbolischen und stochastischen, äusserst effizienten und dynamischen Parsing-Algorithmen, rund um den CKY-Algorithmus, prinzipiell als Ausprägungen eines einzigen Algorithmus, dem sogenannten *Semiring-Parsing-Algorithmus*, aufgefasst werden können. Die unterschiedlichen Leistungen der Parsing-Algorithmen (siehe Abschnitt 2.6), von denen die Fähigkeit zur Disambiguierung wohl die wichtigste ist, resultieren dabei lediglich aus unterschiedlichen Instantiierungen des Semiring-Parsing-Algorithmus. Dies hat mehrere Folgen:

Erstens ist Goodmans Entdeckung zunächst einmal von allgemeinem theoretischen Interesse, da sie zu einem besseren Verständnis der bekannten stochastischen Parsingverfahren für kontextfreie Grammatiken beitragen kann.

Zweitens kann man hoffen, dass in naher Zukunft für kontextfreie Grammatiken ein optimales (für ein gegebenes stochastisches Parsingproblem nicht weiter verbesserbares) stochastisches Parsingverfahren vorhanden sein wird. Denn aufgrund Goodman's Entdeckung darf vermutet werden, dass jedes optimale Parsingverfahren auf der mathematischen Struktur der sogenannten *Semiringe* basiert. Die Suche nach einem optimalen Parsingverfahren scheint also auf die Suche nach einem optimalen Semiring (für das gegebene Parsingproblem) zurückführbar zu sein. Dies ist eine sehr viel leichtere Aufgabe.

Drittens weist Goodman's Entdeckung den Weg zu effizienten Parsingverfahren, da der Semiring-Parsing-Algorithmus geradezu als Prototyp dynamischer Programmierung angesehen werden kann. Unterschiedliche Implementierungen eines beliebigen Parsingverfahrens müssen deshalb nicht mehr auf der Grundlage der Speicher- und Zeitanalyse der gesamten Implementierung bewertet werden, sondern können allein auf der Grundlage der Speicher- und Zeitanalyse der additiven und multiplikativen Operationen des Semirings vorgenommen werden.

Viertens ist Goodman's Entdeckung für die Entwicklung völlig neuer stochastischer Parsingverfahren interessant, da man jeden Semiring prinzipiell als ein neues Parsingverfah-

ren auffassen kann. Relevant sind in diesem Zusammenhang die verschiedenen Gewichtungsmöglichkeiten der Grammatikregeln, sowie die Frage, welche Gewichtungen sich (in annehmbarer Laufzeit) auf die Parsewälder fortsetzen lassen. Am Ende des Abschnitts wird hierzu ein Beispiel vorgestellt.

Fünftens wird die Implementierung vieler verschiedener Parsing-Algorithmen in der einen Gestalt des Semiring-Parsing-Algorithmus durch die modernen objekt-orientierte Programmiersprachen (wie C++, Java, Perl++) optimal unterstützt, weil diese gestatten, die Semiringe als sogenannte Klassen oder Objekte zu implementieren.

Aus diesen Gründen ist der Semiring-Parsing-Algorithmus ein absolutes State-of-The-Art Parsingverfahren.

Da zur Zeit für unifikationsbasierte Grammatiken kein symbolischer Standard-Parsing--Algorithmus bekannt ist, den man für analoge Untersuchungen heranziehen könnte[21], und leider auch kein effizienter stochastischer Parsing-Algorithmus existiert, kann man nicht hoffen, ähnlich vorzügliche Resultate für unifikationsbasierte Grammatiken zu erzielen.

Das State-of-The-Art Parsingverfahren wird also ein Parsingverfahren für kontextfreie (oder einfachere) Grammatiken sein. Vieles deutet daraufhin, dass die sogenannte Lexikalisierung eine wichtige Rolle spielen wird. In Kapitel 5 wird gezeigt, dass probabilistische lexikalisierte kontextfreie Grammatiken ähnlich effizient geparst und trainiert werden können, wie gewöhnliche kontextfreie Grammatiken.

Im folgenden werden Semiringe und der Semiring-Parsing-Algorithmus eingeführt und nachgewiesen, dass die im Abschnitt 2.6 vorgestellten symbolischen und stochastischen Parsing-Algorithmen spezielle Instanzen des Semiring-Parsing-Algorithmus sind. Ferner werden drei weitere stochastische Parsingverfahren direkt als Instanzen des Semiring-Parsing-Algorithmus vorgestellt werden:

- *Viterbi-Parse-Algorithmus*: Berechnung des *Viterbi-Parses*, d.h. der Satzanalyse mit der maximalen Wahrscheinlichkeit,
- *N-Best-Algorithmus*: Berechnung der *N-Best-Wahrscheinlichkeiten*, d.h. der N grössten Wahrscheinlichkeiten aller Satzanalysen,
- *N-Best-Parse-Algorithmus*: Berechnung der *N-Best-Parses*, d.h. aller Satzanalysen, welche die N-Best-Wahrscheinlichkeiten tragen.

Insbesondere der Viterbi-Parse- und der etwas allgemeinere N-Best-Parse-Algorithmus

[21] Der Early-Deduktions-Algorithmus ist fast so etwas wie ein Standardalgorithmus für das Parsen von unifikationsbasierten Grammatiken, wenn man von dem Problem der Unentscheidbarkeit des Parsens mit unifikationsbasierten Grammatiken absieht.

werden sehr oft zu Disambiguierungszwecken herangezogen. Alle drei Algorithmen ähneln dem schon in Abschnitt 2.6 vorgestellten Viterbi-Algorithmus.

In Abschnitt 2.6 wurden für kontextfreie Grammatiken in Chomsky-Normalform sehr detailliert zwei symbolische Parsingverfahren (Parsewald-, CKY-Algorithmus), ein zwischen Symbolik und Stochastik anzusiedelndes Parsingverfahren (Count-Algorithmus), sowie zwei stochastische Parsingverfahren (Inside-Algorithmus, Viterbi-Algorithmus) vorgestellt. Es fällt auf, dass die Prozeduren dieser fünf Parsingverfahren in grossen Teilen übereinstimmen. Die Unterschiede der Algorithmen sind neben unterschiedlicher Ein- und Ausgabe in den folgenden Prozedurzeilen zu lokalisieren:

- **1. Zeile**: Initialisierung der Parsechart mit den Werten $\emptyset$ (Parsewald), $FALSE$ (CKY), 0 (Count, Inside, Viterbi),
- **4. Zeile**: Induktionsanfang, d.h. Gewichtung von $A \Rightarrow^* w_s$ mit den Werten $\{A \rightarrow w_s\}$ (Parsewald), $TRUE$ (CKY), 1 (Count), $p(A \rightarrow w_s)$ (Inside, Viterbi),
- **9.+10. Zeile**: Induktionsschluss, d.h. rekursive Gewichtung von $A \Rightarrow^* w_A$ mittels der Gewichte von $B \Rightarrow^* w_B$ und $C \Rightarrow^* w_C$, sowie (eventuell) eines Gewichts, welches der Grammatikregel $A \rightarrow B\ C$ zugewiesen ist: $p(A \rightarrow B\ C)$ (Inside, Viterbi). Benutzung zweier Rechenoperationen $\vee, \wedge$ (CKY), $+, \cdot$ (Count, Inside), $max, \cdot$ (Viterbi).

Es fällt weiter auf, dass selbst diese Zeilen sehr ähnlich sind. Der in der 9. und 10. Zeile stattfindende Induktionsschluss weist hierbei die grössten Unterschiede auf. Um diese Unterschiede auf eine abstraktere Ebene anzuheben, reicht es, sich klarzumachen, dass am Induktionsschluss die folgenden Elemente beteiligt sind:

- gewisse **Werte** (die den Parsewäldern $A \Rightarrow^* w_A$ rekursiv zugewiesen werden, und den Grammatikregeln $A \rightarrow B\ C$ fest zugewiesen sind),
- eine **Addition**, die mit diesen Werten umgehen kann, sowie
- eine **Multiplikation**, die ebenfalls auf diesen Werten operiert.

Aus der Algebra sind verschiedene mathematische Strukturen $< \mathcal{A}, \oplus, \otimes >$ bekannt, die auf einer abstrakten Menge $\mathcal{A}$ von Elementen, eine abstrakte Addition $\oplus$ und eine abstrakte Multiplikation $\otimes$ erklären. Diese Strukturen heissen *Körper, Ring, Semiring, Gruppe, Ideal,...* abhängig davon, welche Eigenschaften Addition und Multiplikation haben.

Es wird sich erweisen, dass die kontextfreien Parsingverfahren die Eigenschaften sogenannter Semiringe $< \mathcal{A}, \oplus, \otimes, 0, 1 >$ haben. (0 und 1 sind neutrale Elemente der Addition

Elementmenge $\mathcal{A}$: $\{TRUE,\ FALSE\}$ (die Menge der Boolschen Elemente)

Addition $\oplus$: $\vee$ (das logische "oder")

Multiplikation $\otimes$: $\wedge$ (das logische "und")

Additives neutrales Element 0: $FALSE$ (das logische "falsch")

Multiplikatives neutrales Element 1: $TRUE$ (das logische "wahr")

Werte gemäss Grammatik:

$$\mu_{\mathcal{A}}(r) := TRUE \qquad \text{für alle Grammatikregeln } r$$

Abbildung 2.16: CKY-Semiring

Elementmenge $\mathcal{A}$: $\mathcal{N}$ (die Menge der natürlichen Zahlen)

Addition $\oplus$: $+$ (die gewöhnliche Addition)

Multiplikation $\otimes$: $\cdot$ (die gewöhnliche Multiplikation)

Additives neutrales Element 0: 0 (die gewöhnliche Null)

Multiplikatives neutrales Element 1: 1 (die gewöhnliche Eins)

Werte gemäss Grammatik:

$$\mu_{\mathcal{A}}(r) := 1 \qquad \text{für alle Grammatikregeln } r$$

Abbildung 2.17: Count-Semiring

und der Multiplikation, die erst weiter unten definiert werden sollen.) Um dies einzusehen, können mehrere Wege eingeschlagen werden. Der einfachste Weg ist, nicht direkt von einer bestimmten mathematischen Struktur auszugehen, sondern von einem der in Abschnitt 2.6 vorgestellten Parsing-Algorithmen, zum Beispiel von dem CKY-, Count-, Inside-, oder Viterbi-Algorithmus; der Parsewald-Algorithmus ist ungeeignet.

Die in dem ausgewählten Algorithmus benutzte Addition und Multiplikation kann dann gegen eine abstrakte Addition $\oplus$ und eine abstrakte Multiplikation $\otimes$ von abstrakten Chart-Werten $chart_{\mathcal{A}}[.,A,.]$ und Grammatikregel-Werten $\mu_{\mathcal{A}}(A \rightarrow BC)$ aus der abstrakten Menge $\mathcal{A}$ ausgetauscht werden. Die Abbildungen 2.16, 2.17, 2.18, und 2.19 zeigen explizit, welche mathematische Struktur $< \mathcal{A}, \oplus, \otimes, 0, 1 >$ in diesen vier Fällen benötigt wird.

Elementmenge $\mathcal{A}$: $[0,1]$ (das Intervall reeller Zahlen von 0 bis 1)

Addition $\oplus$: $+$ (die gewöhnliche Addition)

Multiplikation $\otimes$: $\cdot$ (die gewöhnliche Multiplikation)

Additives neutrales Element 0: 0 (die gewöhnliche Null)

Multiplikatives neutrales Element 1: 1 (die gewöhnliche Eins)

Werte gemäss Grammatik:

$$\mu_{\mathcal{A}}(r) := p(r) \qquad \text{für alle Grammatikregeln } r$$

Abbildung 2.18: Inside-Semiring

Elementmenge $\mathcal{A}$: $[0,1]$ (das Intervall reeller Zahlen von 0 bis 1)

Addition $\oplus$: max (das Maximum zweier reeller Zahlen)

Multiplikation $\otimes$: $\cdot$ (die gewöhnliche Multiplikation)

Additives neutrales Element 0: 0 (die gewöhnliche Null)

Multiplikatives neutrales Element 1: 1 (die gewöhnliche Eins)

Werte gemäss Grammatik:

$$\mu_{\mathcal{A}}(r) := p(r) \qquad \text{für alle Grammatikregeln } r$$

Abbildung 2.19: Viterbi-Semiring

Eingabe: Ein Semiring $< \mathcal{A}, \oplus, \otimes, 0, 1 >$, eine Grammatik in Chomsky-Normalform, eine Gewichtung $\mu_{\mathcal{A}}(r) \in \mathcal{A}$ der Grammatikregeln r mit Semiring-Elementen, sowie ein Satz $w = w_1 \ldots w_n$.

Ausgabe: Die Semiring-Chart $chart_{\mathcal{A}}[s, A, s+l]$ für alle Teilsätze der Länge $1 \leq l \leq n$, Startindizes $1 \leq s \leq n+1-l$, und Grammatikkategorien A.

Prozedur:

1. $chart_{\mathcal{A}}[.,.,.] := 0$;
2. **for each** $s := 1, \ldots, n$ **do**
3. **for each** $A \rightarrow w_s$ **do**
4. $chart_{\mathcal{A}}[s, A, s+1] := \mu_{\mathcal{A}}(A \rightarrow w_s)$;
5. **for each** $l := 2, \ldots, n$ **do**
6. **for each** $s := 1, \ldots, n+1-l$ **do**
7. **for each** $t := 1, \ldots, l-1$ **do**
8. **for each** $A \rightarrow B\ C$ **do**
9. $chart_{\mathcal{A}}[s, A, s+l] := chart_{\mathcal{A}}[s, A, s+l] \oplus$
10. $\mu_{\mathcal{A}}(A \rightarrow B\ C) \otimes chart_{\mathcal{A}}[s, B, s+t] \otimes$
11. $chart_{\mathcal{A}}[s+t, C, s+l]$;

Abbildung 2.20: Semiring-Parsing-Algorithmus

Wie man sieht, sind sich die Strukturen sehr ähnlich: Vom Count-Semiring kommt man zum Inside-Semiring, indem man die Elementmenge ($\mathcal{N}$ versus $[0,1]$) und die Gewichte der Grammatikregeln r austauscht (1 versus $p(r)$). Vom Inside-Semiring kommt man zum Viterbi-Semiring, indem man die Addition austauscht (max versus +). Viele weitere Beispiele dieser Art wären möglich. Vermutlich ähneln sich die Semiringe aller Parsingverfahren, weil sie alle für die Lösung ähnlicher Probleme konzipiert wurden.

Nach viermaliger Anwendung des Verfahrens hat man zusätzlich zu den vier vorgestellten Semiringen einen abstrakten Parsing-Algorithmus erhalten, welcher in Abbildung 2.20 zu sehen ist und recht treffend *Semiring-Parsing-Algorithmus* genannt werden könnte.

Der Semiring-Parsing-Algorithmus ist ein abstrakter Parsing-Algorithmus, der vor einer Instanziierung mit einem Semiring, weder ein symbolischer noch ein stochastischer Parsing-Algorithmus ist. Wegen der Gewichtung $\mu_{\mathcal{A}}(.)$ der Grammatikregeln ähnelt er allerdings mehr einem stochastischen Parsing-Algorithmus. Während seiner Laufzeit, ist der Semiring-Parsing-Algorithmus natürlich nicht von einem symbolischen

Parsingverfahren ($\mathcal{A}_{CKY}$) oder einem stochastischen Parsingverfahren (beispielsweise $\mathcal{A}_{count}, \mathcal{A}_{inside}, \mathcal{A}_{Viterbi}$) zu unterscheiden.

Die Eingabe des Semiring-Parsing-Algorithmus besteht aus einer Grammatik in Chomsky-Normalform und einem Satz $w = w_1 \dots w_n$, $(n \geq 1)$, sowie einem Semiring $< \mathcal{A}, \oplus, \otimes, 0, 1 >$ und einer Gewichtung $\mu_{\mathcal{A}}(.)$ der Grammatikregeln, die in manchen Fällen aus Regel-Wahrscheinlichkeiten besteht und dann die Eingabe-Grammatik zu einer stochastischen Grammatik macht.

Die Ausgabe des Semiring-Parsing-Algorithmus besteht aus den Elementen der *Semiring-Chart*:

$$chart_{\mathcal{A}}[s, A, s+l] \;:=\; \mu_{\mathcal{A}}(\; A \Rightarrow^* w_s \dots w_{s+l-1} \;) \;.$$

Einem Teilsatz der *Länge* $1 \leq l \leq n$ und dessen *Startindex* $1 \leq s \leq n+1-l$ wird der Wert seines Parsewaldes (der Wurzelkategorie A) zugewiesen, den dieser im Semiring $\mathcal{A}$ besitzt. Hierfür muss garantiert werden, dass die Gewichtung $\mu_{\mathcal{A}}(.)$, die ja zunächst nur für Grammatikregeln definiert ist, sich sinnvoll auf Parsewälder fortsetzen lässt. Die Frage, welche Eigenschaften die abstrakte mathematische Struktur $< \mathcal{A}, \oplus, \otimes, 0, 1 >$ erfüllen muss, damit der Semiring-Parsing-Algorithmus sinnvolle Eigenschaften besitzt, ist deshalb sehr interessant. Die wichtigste Eigenschaft des Semiring-Parsing-Algorithmus sollte sicherlich sein, dass er möglichst viele Parsingverfahren, insbesondere aber den Parsewald-Algorithmus als Instanz besitzt. Für eine Wortfolge w_A mit $|w_A| \geq 2$ sollte daher gelten, dass:

$$
\begin{aligned}
& chart_{\mathcal{A}}(A \Rightarrow^* w_A) \\
& \quad := \mu_{\mathcal{A}}(A \Rightarrow^* w_A) \\
& \quad = \mu_{\mathcal{A}} \left(\sum_{\substack{A \to BC, \\ w_A = w_B w_C}} \; \begin{array}{ccc} & A & \\ & \diagup \quad \diagdown & \\ B \Rightarrow^* w_B & & C \Rightarrow^* w_C \end{array} \right) \\
& \quad = \sum_{\substack{A \to BC, \\ w_A = w_B w_C}} \mu_{\mathcal{A}} \left(\begin{array}{ccc} & A & \\ & \diagup \quad \diagdown & \\ B \Rightarrow^* w_B & & C \Rightarrow^* w_C \end{array} \right)
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{\substack{A \to BC, \\ w_A = w_B w_C}} \mu_{\mathcal{A}}(A \to B\ C) \otimes \mu_{\mathcal{A}}(B \Rightarrow^* w_B) \otimes \mu_{\mathcal{A}}(C \Rightarrow^* w_C) \\
&= \sum_{\substack{A \to BC, \\ w_A = w_B w_C}} \mu_{\mathcal{A}}(A \to B\ C) \otimes chart_{\mathcal{A}}(B \Rightarrow^* w_B) \otimes chart_{\mathcal{A}}(C \Rightarrow^* w_B)
\end{aligned}
$$

Offensichtlich wird diese Rekursion in der 9.-11. Zeile der Prozedur des Semiring-Parsing-Algorithmus realisiert. Alle weiteren Zeilen der Prozedur sind klar: In der 1. Zeile des Semiring-Parsing-Algorithmus wird die Semiring-Chart mit dem additiv neutralen Element 0 initialisiert, während in der 2.-4. Zeile der Induktionsanfang und in der 5.-11. Zeile der Induktionsschluss stattfindet. Interessant ist noch die Benutzung der Werte $\mu_{\mathcal{A}}(A \to a)$ in der 4. Zeile.

Der Semiring-Parsing-Algorithmus ist von der Ordnung $O(n^3)$, wenn die Addition und Multiplikation zweier Elemente des Semirings in konstanter Zeit vorgenommen werden können, was bei allen vier vorgestellten Semiringen der Fall ist.

Im folgenden soll gelöst werden, welche Eigenschaften die abstrakte Addition $\oplus$ und die abstrakte Multiplikation $\otimes$ haben müssen, um im Semiring-Parsing-Algorithmus sinnvoll eingesetzt werden zu können. Wie bereits gezeigt wurde, muss für die rekursive Berechnung von $chart_{\mathcal{A}}(C \Rightarrow^* w_B)$ der äquivalente Ausdruck

$$
\sum_{\substack{A \to BC, \\ w_A = w_B w_C}} \mu_{\mathcal{A}}(A \to B\ C) \otimes chart_{\mathcal{A}}(B \Rightarrow^* w_B) \otimes chart_{\mathcal{A}}(C \Rightarrow^* w_B)
$$

Sinn machen. Man liest diesem Ausdruck ab, dass die Addition $\oplus$ kommutativ und assoziativ sein muss (sonst ist die Notation $\sum$ sinnlos), die Multiplikation $\otimes$ muss hingegen nur assoziativ sein (sonst macht der ungeklammerte Ausdruck $\mu_{\mathcal{A}}(A \to B\ C) \otimes chart_{\mathcal{A}}(B \Rightarrow^* w_B) \otimes chart_{\mathcal{A}}(C \Rightarrow^* w_B)$ keinen Sinn). Ferner muss die Multiplikation ein neutrales Element besitzen (weil $\mu_{\mathcal{A}}(A \to B\ C)$ in manchen Parsingverfahren konstant 1 ist). Ebenso muss die Addition ein neutrales Element besitzen (weil die Summe $\sum$ mit 0 initialisiert werden muss und auch weil ein Summand $\mu_{\mathcal{A}}(A \to B\ C) \otimes chart_{\mathcal{A}}(B \Rightarrow^* w_B) \otimes chart_{\mathcal{A}}(C \Rightarrow^* w_B)$ manchmal 0 sein kann).

Die Abbildung 2.21 fasst die erforderlichen Eigenschaften zusammen. Es sind die Eigenschaften eines Semirings, die man in jedem Buch über Algebra nachschlagen kann, zum Beispiel in (Van der Warden 1936). Erwähnenswert ist, dass die Multiplikation eines Semirings nicht kommutativ sein muss, was den Semiring zu einem Ring machen würde.

Elementmenge $\mathcal{A}$: beliebig

Addition $\oplus$:

$$\begin{array}{lll} a \oplus b = b \oplus a & \forall a, b \in \mathcal{A} & \text{(Kommutativität)} \\ (a \oplus b) \oplus c = a \oplus (b \oplus c) & \forall a, b, c \in \mathcal{A} & \text{(Assoziativität)} \end{array}$$

Multiplikation $\otimes$:

$$\begin{array}{lll} (a \otimes b) \otimes c = a \otimes (b \otimes c) & \forall a, b, c \in \mathcal{A} & \text{(Assoziativität)} \\ a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c) & \forall a, b, c \in \mathcal{A} & \text{(Distributivität)} \\ a \otimes 0 = 0 = 0 \otimes a & \forall a \in \mathcal{A} & \end{array}$$

Additives neutrales Element 0:

$$a \oplus 0 = a = 0 \oplus a \quad \forall a \in \mathcal{A}$$

Multiplikatives neutrales Element 1:

$$a \otimes 1 = a = 1 \otimes a \quad \forall a \in \mathcal{A}$$

Werte gemäss Grammatik:

$$\mu_{\mathcal{A}} : \; G \to \mathcal{A}, \quad r \mapsto \mu_{\mathcal{A}}(r) \; .$$

Abbildung 2.21: Eigenschaften eines Semirings

Elementmenge $\mathcal{A}$:

$$\left\{ X \subseteq \mathcal{T}(G) \;\middle|\; \begin{array}{l} X \text{ ist leer oder von der Gestalt:} \\ \{A \to B\ C\},\ X_B, \\ \begin{array}{cc} A & \\ X_B & C \end{array} \text{ oder } \begin{array}{cc} ? & \\ X_B & X_C \end{array} \end{array} \right\}$$

Hierbei bezeichnet $\{A \to B\ C\}$ einen aus einer einzigen verzweigenden Grammatikregel bestehenden Parsewald, X_B eine Teilmenge der Syntaxbäume eines festen (aber beliebigen) Teilsatzes w_B (d.h. $X_B \subseteq B \Rightarrow^* w_B$), sowie:

$$\begin{array}{cc} A & \\ X_B & C \end{array} := \left\{ \begin{array}{cc} A & \\ x_B & C \end{array} \;\middle|\; x_B \in X_B \right\} \text{ und } \begin{array}{cc} ? & \\ X_B & X_C \end{array} := \langle X_B, X_C \rangle$$

Addition $\oplus$: wird (im Semiring-Parsing-Algorithmus) nur für disjunkte $X_1,\ X_2 \subset \mathcal{T}(G)$ der Gestalt X_A benötigt:

$$X_1 \oplus X_2 := X_1 \cup X_2\ .$$

Multiplikation $\otimes$: wird (im Semiring-Parsing-Algorithmus) nur für die folgenden vier Fälle benötigt:

$$\{A \to B\ C\} \otimes X_B := \begin{array}{cc} A & \\ X_B & C \end{array} \text{ und } \begin{array}{cc} A & \\ X_B & C \end{array} \otimes X_C := \begin{array}{cc} A & \\ X_B & X_C \end{array}$$

sowie:

$$X_B \otimes X_C := \begin{array}{cc} ? & \\ X_B & X_C \end{array} \text{ und } \{A \to B\ C\} \otimes \begin{array}{cc} ? & \\ X_B & X_C \end{array} := \begin{array}{cc} \mathbf{A} & \\ X_B & X_C \end{array}$$

Wie man leicht sieht, ist die so definierte Multiplikation wohldefiniert, abgeschlossen, assoziativ, aber nicht kommutativ.

Additives neutrales Element 0: $\emptyset$ (die leere Menge)

Multiplikatives neutrales Element 1: die den "leeren Syntaxbaum" enthaltende Menge. Wird (im Semiring-Parsing-Algorithmus) nicht benötigt.

Werte gemäss Grammatik:

$$\mu_{\mathcal{A}}(r) := \{r\} \qquad \text{für alle Grammatikregeln } r$$

Abbildung 2.22: Parsewald-Semiring

Im folgenden werden vier weitere Semiringe vorgestellt (Parsewald-, Viterbi-Parse, N-Best-, N-Best-Parse-Semiring), die sich in Goodman (1998) ebenfalls finden lassen. Bis auf den Viterbi-Parse-Semiring werden in dieser Arbeit im Vergleich zu Goodman's Arbeit abweichende, weil einfachere und genauere, Angaben gemacht. Dies betrifft vor allem den Parsewald-Semiring, der in Abbildung 2.22 angegeben ist und der bei Goodman nur sehr schwer zu verstehen ist.

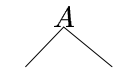
Der Parsewald-Semiring instantiiert den Parsewald-Algorithmus (siehe Abbildung 2.11). Die Addition und Multiplikation des Parsewald-Semirings wird nur für die im Semiring--Parsing-Algorithmus vorkommenden Elemente eingeführt. Der Schlüsselschritt ist, dass erkannt wird, dass im Semiring-Parsing-Algorithmus zur Bildung des Produkts $\mu_{\mathcal{A}}(A \to BC) \otimes chart_{\mathcal{A}}(B \Rightarrow^* w_B) \otimes chart_{\mathcal{A}}(C \Rightarrow^* w_C)$, kürzer als

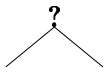
$$\{A \to BC\} \otimes X_B \otimes X_C$$

notiert, die folgenden vier Typen von Semiring-Elementen zur Verfügung stehen müssen (je nachdem in welcher Reihenfolge das Produkt berechnet wird):

$$\{A \to B\ C\}, \quad X_B, \quad \{A \to B\ C\} \otimes X_B, \quad X_B \otimes X_C\ .$$

In Abbildung 2.22 findet man diese vier Typen in etwas sinnvollerer Notation:

$\{A \to B\ C\}$, X_B, 

, 

.

Bemerkenswert ist, dass die Multiplikation des Parsewald-Semirings nicht kommutativ ist.

Die Abbildung 2.23 zeigt den Semiring zur Bestimmung der Viterbi-Parses. Da es natürlich mehr als einen Parse maximaler Wahrscheinlichkeit geben kann, muss zu jeder Viterbi-Wahrscheinlichkeit ein ganzer Parsewald verwaltet werden. In den meisten Implementierungen von stochastischen Parsern wird darauf allerdings verzichtet und (mittels Zufallsauswahl) lediglich ein Viterbi-Parse verwaltet. Leider zerstört ein derartiges Vorgehen die Assoziativität der Addition, sodass diese Implementatierungen mangelhaft sind.

Recht hübsch ist der Rückgriff auf Addition und Multiplikation des Viterbi- und Parsewald-Semirings, was die Verständlichkeit der gezeigten Operationen stark erhöht.

Abschliessend werden zwei Parsing-Algorithmen vorgestellt, die den Viterbi- und den Viterbi-Parse-Algorithmus verallgemeinern.

Mit dem N-Best-Algorithmus sollen die N-Best-Wahrscheinlichkeiten, d.h. die N grössten Wahrscheinlichkeiten aller Parses, bestimmt werden. Mit dem N-Best-Parse-Algorithmus werden die N-Best-Parses, d.h. die dazugehörigen Parses, ermittelt.

Elementmenge $\mathcal{A}$: $\mathcal{A}_{Viterbi} \times \mathcal{A}_{Parsewald}$

Addition $\oplus$:

$$< p, X_p > \oplus < q, X_q > := \begin{cases} < p \oplus_{Viterbi} \ q, \ X_p \oplus_{Parsewald} \ X_q > & \text{falls } p = q \\ < p \oplus_{Viterbi} \ q, \ X_{p \oplus_{Viterbi} \ q} > & \text{falls } p \neq q \end{cases}$$

Multiplikation $\otimes$:

$$< p, X_p > \otimes < q, X_q > := < p \otimes_{Viterbi} q, \ X_p \otimes_{Parsewald} X_q >$$

Additives neutrales Element 0: $< 0_{Viterbi}, 0_{Parsewald} >$

Multiplikatives neutrales Element 1: $< 1_{Viterbi}, 1_{Parsewald} >$

Werte gemäss Grammatik:

$$\mu_{\mathcal{A}}(r) := < \mu_{\mathcal{A}Viterbi}(r), \ \mu_{\mathcal{A}Parsewald}(r) > \text{ für alle Grammatikregeln } r$$

Abbildung 2.23: Viterbi-Parse-Semiring

Elementmenge $\mathcal{A}$:

$$\{M \subset \mathcal{A}_{Viterbi} \mid 1 \leq |M| \leq N\}$$

Addition $\oplus$: die gewöhnliche Mengenvereinigung, gefolgt von der Extraktion der N grössten Zahlen:

$$M_1 \oplus M_2 := N\text{–best}(M_1 \cup M_2)$$

Multiplikation $\otimes$:

$$M_1 \otimes M_2 := N\text{–best}\{m_1 \otimes_{Viterbi} m_2 \mid m_1 \in M_1, \ m_2 \in M_2\}$$

Additives neutrales Element 0: $\{0_{Viterbi}\}$

Multiplikatives neutrales Element 1: $\{1_{Viterbi}\}$

Werte gemäss Grammatik:

$$\mu_{\mathcal{A}}(r) := \{ \ \mu_{\mathcal{A}Viterbi}(r) \ \} \text{ für alle Grammatikregeln } r$$

Abbildung 2.24: N-Best-Semiring ($N = 1$ (Viterbi-Semiring), $2, 3, \ldots$)

Ein einfaches Beispiel zeigt, dass die formale Definition von "N-Best" recht unangenehm ist. Sind beispielsweise $<0.4, x_1>$, $<0.4, x_2>$, $<0.05, x_3>$, $<0.05, x_4>$, $<0.05, x_5>$, $<0.03, x_6>$, $<0.02, x_7>$ die sieben Parses eines gegebenen Satzes, zusammen annotiert mit ihren Analyse-Wahrscheinlichkeiten, so ist die 1-Best-Wahrscheinlichkeit (d.h. die Viterbi-Wahrscheinlichkeit) gleich 0.4, die 2-Best-Wahrscheinlichkeiten sind $\{0.04, 0.05\}$, und die 3-Best-Wahrscheinlichkeiten sind $\{0.04, 0.05, 0.03\}$. Somit gibt es zwei 1-Best-Parses (Viterbi-Parses) $\{x_1, x_2\}$, sogar fünf 2-Best-Parses $\{x_1, \ldots, x_5\}$, sowie sechs 3-Best-Parses $\{x_1, \ldots, x_6\}$.

Der einfachste Weg zu einer formalen Definition eines $N-$best(.)-Operators führt über Ordnungen. Sind

$$<p_1, X_1> \ldots <p_1, X_{i_1}>, <p_2, X_{1+i_1}> \ldots <p_2, X_{i_2}>, <p_3, X_{1+i_2}> \ldots <p_3, X_{i_3}> \ldots$$

die Elemente einer endlichen Menge M von 2-dimensionalen Daten, die so angeordnet sind, dass in der ersten Koordinate die folgende Kette von Ungleichungen gilt:

$$p_1 > p_2 > p_3 > \ldots \ ,$$

so gilt $i_1 < i_2 < i_3 < \ldots \leq |M|$, sowie:

$$N\text{–best}(M) := \{\ <p_i, X_i> \in M \mid i \leq i_N\ \} \ .$$

Für eine Menge M von 1-dimensionalen Daten, d.h. für Zahlenmengen, erhält man natürlich analog:

$$i_1 = 1,\ i_2 = 2,\ i_3 = 3, \ldots \ .$$

Dieser wichtige Spezialfall besagt, dass N–best(.) aus einer Zahlenmenge deren N grössten Zahlen extrahiert:

$$N\text{–best}(M) := \{\ p_i \in M \mid i \leq N\ \} \ .$$

Offensichtlich kann hier gefolgert werden, dass $|N\text{–best}(M)| \leq N$. Es wurde bereits per Beispiel gezeigt, dass dies im allgemeinen nicht gelten wird.

Der $N-$best-Operator ist der Schlüssel zu beiden Semiringen, dem N-Best- und dem N-Best-Parse-Semiring. Bei einer Implementierung sollte darauf geachtet werden, dass der N–best-Operator in konstanter Zeit arbeitet, was für Zahlenmengen einfach zu realisieren ist, aber für Mengen von 2-dimensionalen Daten schwieriger erscheint.

Die Abbildung 2.24 zeigt den N-Best-Semiring. Mit diesem Semiring liegt eine Verallgemeinerung des Viterbi-Semirings vor, den man für $N = 1$ erhält. Natürlich ist es sinnvoll bei der Definition des N-Best-Semirings auf den Viterbi-Semiring zurückzugreifen: Die

Elementmenge $\mathcal{A}$:

$$\{M \subset \mathcal{A}_{Viterbi-Parse} \mid 1 \leq |M| \leq N\}$$

Addition $\oplus$: die gewöhnliche Mengenvereinigung, gefolgt von der $N-$best(.)-Extraktion:

$$M_1 \oplus M_2 := N\text{–best}(M_1 \cup M_2)$$

Multiplikation $\otimes$:

$$M_1 \otimes M_2 := N\text{–best}\{m_1 \otimes_{Viterbi-Parse} m_2 \mid m_1 \in M_1,\ m_2 \in M_2\}$$

Additives neutrales Element 0: $\{0_{Viterbi-Parse}\}$

Multiplikatives neutrales Element 1: $\{1_{Viterbi-Parse}\}$

Werte gemäss Grammatik:

$$\mu_{\mathcal{A}}(r) := \{\ \mu_{\mathcal{A}Viterbi-Parse}(r)\ \} \text{ für alle Grammatikregeln } r$$

Abbildung 2.25: N-Best-Parse-Semiring ($N = 1$ (Viterbi-Parse-Semiring), $2, 3, \ldots$)

Elemente des N-Best-Semiring sind die 1- bis N-elementigen Teilmengen des Viterbi-Semirings. Die Addition besteht aus der gewöhnlichen Mengenvereinigung, die Multiplikation aus einer kreuzweisen Produktbildung, beide Operationen gefolgt von der Anwendung des N–best-Operators, der in diesem Fall die N grössten Zahlen extrahiert.

Die Abbildung 2.25 zeigt den N-Best-Parse-Semiring. Mit diesem Semiring liegt eine Verallgemeinerung des Viterbi-Parse-Semirings vor, den man für $N = 1$ erhält. Natürlich ist es sinnvoll, bei der Definition des N-Best-Parse-Semirings auf den Viterbi-Parse-Semiring zurückzugreifen: Die Elemente des N-Best-Parse-Semirings sind die 1 bis N-elementigen Teilmengen des Viterbi-Parse-Semirings. Die Addition besteht aus der gewöhnlichen Mengenvereinigung, die Multiplikation hingegen aus einer kreuzweisen Produktbildung, für welche natürlich die Multiplikation des Viterbi-Parse-Semirings benutzt wird. Beide Operationen werden durch die Anwendung des N–best-Operators vervollständigt, der in diesem Fall die Parsewälder aller N N-Best-Wahrscheinlichkeiten extrahiert.

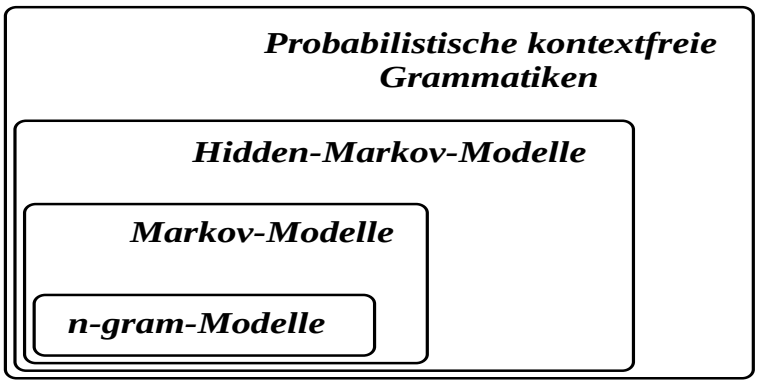


Abbildung 2.26: Sprachmodellierung und probabilistische kontextfreie Grammatiken

2.8 Sprachmodellierung mit kontextfreien Grammatiken

Die Estimierung von Wahrscheinlichkeitsverteilungen der verschiedensten Sprachphänomene und diese in Sprachanwendungen einzusetzen kann als Aufgabe der statistischen Sprachmodellierung angesehen werden. Neben ihrem Ursprungsgebiet, der Spracherkennung (Jelinek 1997), werden Sprachmodelle sowohl für klassische und linguistisch komplexe Aufgaben, wie die maschinelle Übersetzung (Brown et al. 1990), als auch für moderne Aufgaben mit eher kleinem linguistischen Anforderungsprofil, wie dem Information-Retrieval (Berger und Lafferty 1999), eingesetzt. Rosenfeld (2000) gibt einen umfassenden Überblick über die Anwendungen und die vielfältigen Methoden aus der Sprachmodellierung während der letzten 20 Jahre.

Jede Wahrscheinlichkeitsverteilung über die möglichen Sätze (oder über akustische Signale, oder Dokumente, oder andere linguistische Einheiten) könnte prinzipiell als ein *statistisches Sprachmodell* bezeichnet werden. Natürlich ist diese Definition viel zu allgemein, da auf diese Weise keine Abgrenzung gegenüber stochastischen Grammatiken möglich ist, die ja auch allen grammatischen Sätzen eine Wahrscheinlichkeit zuweisen. Wahrscheinlich ist es daher legitim, die *n-gram-Modelle*, oder allgemeiner, die *Markov-Modelle*, und noch allgemeiner, die *Hidden Markov-Modelle* als typische Vertreter der Sprachmodelle anzusehen. Diese wurden in den 80er Jahren innerhalb der Computer-Linguistik als erste statistische Modelle angewandt, und schon in den späten 60er bzw. frühen 70er Jahren mathematisch erforscht. Ohne Zweifel sind die Hidden Markov-Modelle wegen ihrer langen Geschichte die populärsten Modelle innerhalb der stochastischen Linguistik. Sprachmodelle haben im allgemeinen einen positiven Einfluss auf die Anwendung, in der sie eingesetzt werden. Trotzdem sind sie im Vergleich mit probabilistischen kontextfreien Grammatiken möglicherweise nicht optimal:

- Sprachmodelle reagieren oft negativ bei einer Änderung von *Stil, Thema* und *Genre*

des Textes auf dem sie trainiert wurden.

- Es ist wenig erforscht, ob Sprachmodelle durch die Nutzung linguistischen Vorwissens, wie zum Beispiel semantischer Taxonomien (WordNet, GermaNet), aber auch manuell erstellte Grammatiken, verbessert werden können.

Während es einige Arbeiten zu klassen-basierten Sprachmodellen (Brown et al. (1992), Kneser und Ney (1993), Saul und Pereira (1997)) und auch zu syntax-basierten Sprachmodellen (Della Pietra et al. (1994), Chelba (1997), Chelba et al. (1997), Chelba und Jelinek (1998), Chelba und Jelinek (2000), Khudanpur und Wu (2000), Benedi und Sanchez (2000)) vorhanden sind, sind dem Autor keine Arbeiten zu Sprachmodellen bekannt, die ein semantisches Vorwissen ausnutzen.

Im folgenden werden n-gram-Modelle, Markov-Modelle und Hidden-Markov-Modelle vorgestellt. Der Abbildung 2.26 entnimmt man, dass von den Sprachmodellen die Hidden-Markov-Modelle die allgemeinsten Modelle sind: n-gram-Modelle können als spezielle Markov-Modelle, und diese als spezielle Hidden-Markov-Modelle aufgefasst werden. Das Ziel dieses Abschnitts ist es, zu zeigen, dass die Hidden Markov-Modelle ihrerseits als spezielle, sehr einfache probabilistische kontextfreie Grammatiken angesehen werden können. In Abbildung 2.26 wird auch diese Beziehung dargestellt. Das Training und die Anwendung von Hidden-Markov-Modellen muss daher nicht explizit vorgestellt werden, sondern erschliesst sich aus den Trainingsverfahren (Abschnitte 2.5 und 3.1) und den Anwendungen (Abschnitte 2.6 und 2.7) für probabilistische kontextfreie Grammatiken. Wegen dieses engen Zusammenhangs ist zu vermuten, dass sich erzielte Innovationen auf dem Gebiet probabilistischer kontextfreier Grammatiken für Hidden Markov-Modelle ebenfalls positiv auswirken könnten. Da probabilistische kontextfreie Grammatiken in den letzten Jahren in einem verstärkten Forschungs-Interesse standen, hofft der Autor, dass aus dieser neuen Sichtweise bessere Sprachmodelle resultieren werden.

Fast alle Sprachmodelle zerlegen die Wahrscheinlichkeit eines Satzes in ein Produkt von bedingten Wahrscheinlichkeiten (Rosenfeld 2000). In dieser Arbeit wird deshalb die folgende Wahrscheinlichkeitsverteilung als *Sprachmodell* bezeichnet:

$$p(s) \;:=\; p(w_1 \ldots w_T) \;=\; p(w_1|h_1) \cdots p(w_T|h_T) \;=\; \prod_{t=1}^{T} p(w_t|h_t) \;.$$

Hierbei ist w_t das t-te Wort im Satz und

$$h_t := h(w_1 \ldots w_{t-2} w_{t-1})$$

dessen *Geschichte*. Die Position t des Wortes im Satz wird gelegentlich auch als *Zeitpunkt* bezeichnet. Hinter diesen beiden Bezeichnungen steht offenbar die Vorstellung, dass ein

$$
\begin{aligned}
p(w_1 \ldots w_T) &= p(w_1) \cdot p(w_2|w_1) \cdot p(w_3|w_1 w_2) \cdots p(w_T|w_{T-2} w_{T-1}) \\
&= p(w_1 w_2 w_3) \cdot p(w_4|w_2 w_3) \cdots p(w_T|w_{T-2} w_{T-1}) \qquad (\text{für } T \geq 3), \\
p(w_1 \ldots w_T) &= p(w_1) \cdot p(w_2|w_1) \cdots p(w_T|w_{T-1}) \\
&= p(w_1 w_2) \cdot p(w_3|w_2) \cdots p(w_T|w_{T-1}) \qquad (\text{für } T \geq 2).
\end{aligned}
$$

Abbildung 2.27: Trigram-Modell (oben) und Bigram-Modell (unten)

Satz nicht auf einmal, sondern inkrementell, d.h. wortweise (jedes Wort zu seinem Zeitpunkt), von links nach rechts, generiert wird. Die angegebenen Formeln sollen ausdrücken, dass ein Sprachmodell jedes Wort eines Satzes höchstens in Abhängigkeit der vorhergehenden Worte generiert. Die Definition der Sprachmodelle soll mit der Bemerkung abgerundet werden, dass es kurioserweise keine linguistisch motivierte Definition von dem gibt, was ein Sprachmodell als Satz benutzt. Es scheint so, dass Sprachmodelle in dieser Beziehung nicht so trainiert werden, wie sie angewendet und evaluiert werden: Die Parameter von Sprachmodellen werden im allgemeinen auf einem grossen Textkorpus estimiert, indem dieser Textkorpus als ein einziger langer Satz angesehen wird (siehe unten: Forward-Backward-Algorithmus). Evaluiert werden Sprachmodelle dagegen in der Regel durch Messung der Korpusperplexität (siehe Abschnitt 2.3), offensichtlich also unter Verwendung eines Korpus von linguistischen Sätzen und der Wahrscheinlichkeiten dieser Sätze, so wie sie das trainierte Sprachmodell ausgibt (Rosenfeld 2000).[22] Auf dem Web stehen mehrere, teilweise recht umfangreiche, für Forschungszwecke frei verfügbare, Software-Pakete zum Download bereit (McCallum (1996), Clarkson und Rosenfeld (1997), Stolcke (1999)).

Nachfolgend werden *n-gram-Modelle* als spezielle Markov-Modelle charakterisiert werden. Definiert sind *n-gram-Modelle* jedoch als Sprachmodelle mit der Geschichte:

$$h_t :=< w_{t-(n-1)} \ldots w_{t-2} w_{t-1} >, \qquad (t = 1 \ldots T) \ .$$

Dies bedeutet, dass in einem n-gram-Modell jedes Wort in Abhängigkeit der höchstens $n-1$ vorhergehenden Worte generiert wird. In der Regel wird angenommen, dass $n \geq 2$ gilt. Das *Unigram-Modell* ($n = 1$) ist natürlich insofern uninteressant, da es die gerade

[22] Vermutlich ist es daher in der Sprachmodellierung üblich, spezielle Satzanfang- und Satzende-Markierungen für die linguistischen Sätze zu benutzen, indem etwa neue Worte *BeginOfSentence* und *EndOfSentence* am Anfang und Ende eines linguistischen Satzes eingefügt werden. Mit diesem Trick kann die bestehende Inkonsistenz zwischen Training und Anwendung der meisten Sprachmodelle gemildert werden. Zum Beispiel tragen für Bigram-Modelle die Wahrscheinlichkeiten $p(w|BeginOfSentence)$ und $p(EndOfSentence|w)$ dazu bei, dass linguistische Sätze stochastisch angemessen modelliert werden.

angesprochenen Abhängigkeiten nicht modelliert. Die Abbildung 2.27 zeigt die zwei populärsten Sprachmodelle, das *Bigram-Modell* ($n = 2$) und das *Trigram-Modell* ($n = 3$). Die Bigram-Modelle werden in der Praxis sicherlich am häufigsten benutzt. Ein Trigram-Modell ist nur dann sinnvoll anwendbar, wenn es auf einem sehr grossen Trainingskorpus (einige 10 Millionen Wörter) trainiert wurde. Die bedingten Wahrscheinlichkeiten der n-gram-Modelle sind wie folgt definiert:

$$p(w|h) := \frac{p(hw)}{p(h)} \ .$$

Hierbei ist die $(n-1)$-dimensionale Wahrscheinlichkeitsverteilung $p(h)$ als *Randverteilung* einer vorgegebenen, aber beliebigen, n-dimensionalen Wahrscheinlichkeitsverteilung $p(hw)$ definiert:

$$p(h) := \sum_w p(hw)$$

Damit kann leicht verifiziert werden, dass die Wahrscheinlichkeiten $p(w|h)$ bei festgehaltener Geschichte h eine Wahrscheinlichkeitsverteilung über die Menge aller Worte w bilden:

$$\sum_w p(w|h) = \sum_w \frac{p(hw)}{p(h)} = \frac{\sum_w p(hw)}{p(h)} = \frac{p(h)}{p(h)} = 1 \ .$$

Hieraus folgt, dass durch ein n-gram-Modell eine Wahrscheinlichkeitsverteilung über die Menge aller Sätze mit fester Länge T gegeben ist[23]:

$$\begin{aligned} \sum_{w_1 \ldots w_T} p(w_1 \ldots w_T) &= \sum_{w_1 \ldots w_T} p(w_1|h_1) \cdots p(w_T|h_T) \\ &= \sum_{w_1} p(w_1|h_1) \ldots \sum_{w_T} p(w_T|h_T) \\ &= 1 \ . \end{aligned}$$

In einem n-gram Modell gilt insbesondere (per Definition):

$$p(w_1 \ldots w_n) = \prod_{t=1}^{n} p(w_t|h_t) \ .$$

Für Wortfolgen der Länge $T > n$ gilt daher:

$$p(w_1 \ldots w_T) = p(w_1 \ldots w_n) \cdot \prod_{t=n+1}^{T} p(w_t|h_t) \ . \tag{2.5}$$

Die Wortfolgen der Länge n:

$$h_t w_t = w_{t-(n-1)} \ldots w_{t-1} w_t \ ,$$

die für $t \geq n$ im Zähler der Definition von $p(w_t|h_t)$ vorkommen, werden *n-gram-Datentoken* genannt, und geben den n-gram-Modellen ihren Namen.

with certain physical (0.00684) will actually be (0.00684) which will actually (0.00684) when one message (0.00684) unknown at the (0.00684) to the engineering (0.00684) to some system (0.00684) to or are (0.00684) to operate for (0.00684) time of design (0.00684) this number ((0.00684) this is unknown (0.00684) they refer to (0.00684) then this number (0.00684) the time of (0.00684) the set , (0.00684) the one which (0.00684) the number of (0.00684) the messages have (0.00684) the engineering problem (0.00684) the actual message (0.00684) that the actual (0.00684) that of reproducing (0.00684) that is they (0.00684) system with certain (0.00684) system must be (0.00684) some system with (0.00684) since this is (0.00684) significant aspect is (0.00684) set of possible (0.00684) set , all (0.00684) semantic aspects of (0.00684) selection , not (0.00684) selected from a (0.00684) selected at another (0.00684) reproducing at one (0.00684) regarded as a (0.00684) refer to or (0.00684) produced when one (0.00684) problem of communication (0.00684) problem . The (0.00684) possible selection , (0.00684) possible messages . (0.00684) point either exactly (0.00684) point . Frequently (0.00684) physical or conceptual (0.00684) or conceptual entities (0.00684) or are correlated (0.00684) or approximately a (0.00684) operate for each (0.00684) one which will (0.00684) one selected from (0.00684) one point either (0.00684) one message is (0.00684) of reproducing at (0.00684) of possible messages (0.00684) of messages is (0.00684) of information produced (0.00684) of design . (0.00684) of communication is (0.00684) of communication are (0.00684) number of messages (0.00684) number (. (0.00684) not just the (0.00684) must be designed (0.00684) messages is finite (0.00684) messages have meaning (0.00684) messages . The (0.00684) message selected at (0.00684) message is one (0.00684) message is chosen (0.00684) measure of information (0.00684) meaning ; that (0.00684) just the one (0.00684) is unknown at (0.00684) is they refer (0.00684) is that the (0.00684) is that of (0.00684) is one selected (0.00684) is finite then (0.00684) is chosen from (0.00684) irrelevant to the (0.00684) information produced when (0.00684) have meaning ; (0.00684) fundamental problem of (0.00684) from the set (0.00684) from a set (0.00684) for each possible (0.00684) finite then this (0.00684) exactly or approximately (0.00684) equally likely . (0.00684) entities . These (0.00684) engineering problem . (0.00684) either exactly or (0.00684) each possible selection (0.00684) designed to operate (0.00684) design . If (0.00684) correlated according to (0.00684) conceptual entities . (0.00684) communication is that (0.00684) communication are irrelevant (0.00684) chosen since this (0.00684) chosen from the (0.00684) choices being equally (0.00684) certain physical or (0.00684) can be regarded (0.00684) being equally likely (0.00684) be regarded as (0.00684) be designed to (0.00684) be chosen since (0.00684) at the time (0.00684) at one point (0.00684) at another point (0.00684) aspects of communication (0.00684) aspect is that (0.00684) as a measure (0.00684) are irrelevant to (0.00684) are correlated according (0.00684) approximately a message (0.00684) another point . (0.00684) all choices being (0.00684) actually be chosen (0.00684) actual message is (0.00684) according to some (0.00684) a set of (0.00684) a message selected (0.00684) a measure of (0.00684) These semantic aspects (0.00684) The system must (0.00684) The significant aspect (0.00684) The fundamental problem (0.00684) If the number (0.00684) Frequently the messages (0.00684) ; that is (0.00684) . These semantic (0.00684) . The system (0.00684) . The significant (0.00684) . If the (0.00684) . Frequently the (0.00684) . . . (0.00684) . .) (0.00684) .) can (0.00684) , not just (0.00684) , all choices (0.00684)) can be (0.00684) (. . (0.00684)

Abbildung 2.28: Die empirischen Wahrscheinlichkeiten der Trigram-Datentypen aus Shannons Zitat

of communication (0.01360) message is (0.01360) is that (0.01360) . The (0.01360) . . (0.01360) with certain (0.00680) will actually (0.00680) which will (0.00680) when one (0.00680) unknown at (0.00680) to the (0.00680) to some (0.00680) to or (0.00680) to operate (0.00680) time of (0.00680) this number (0.00680) this is (0.00680) they refer (0.00680) then this (0.00680) the time (0.00680) the set (0.00680) the one (0.00680) the number (0.00680) the messages (0.00680) the engineering (0.00680) the actual (0.00680) that the (0.00680) that of (0.00680) that is (0.00680) system with (0.00680) system must (0.00680) some system (0.00680) since this (0.00680) significant aspect (0.00680) set of (0.00680) set , (0.00680) semantic aspects (0.00680) selection , (0.00680) selected from (0.00680) selected at (0.00680) reproducing at (0.00680) regarded as (0.00680) refer to (0.00680) produced when (0.00680) problem of (0.00680) problem . (0.00680) possible selection (0.00680) possible messages (0.00680) point either (0.00680) point . (0.00680) physical or (0.00680) or conceptual (0.00680) or are (0.00680) or approximately (0.00680) operate for (0.00680) one which (0.00680) one selected (0.00680) one point (0.00680) one message (0.00680) of reproducing (0.00680) of possible (0.00680) of messages (0.00680) of information (0.00680) of design (0.00680) number of (0.00680) number ((0.00680) not just (0.00680) must be (0.00680) messages is (0.00680) messages have (0.00680) messages . (0.00680) message selected (0.00680) measure of (0.00680) meaning ; (0.00680) likely . (0.00680) just the (0.00680) is unknown (0.00680) is they (0.00680) is one (0.00680) is finite (0.00680) is chosen (0.00680) irrelevant to (0.00680) information produced (0.00680) have meaning (0.00680) fundamental problem (0.00680) from the (0.00680) from a (0.00680) for each (0.00680) finite then (0.00680) exactly or (0.00680) equally likely (0.00680) entities . (0.00680) engineering problem (0.00680) either exactly (0.00680) each possible (0.00680) designed to (0.00680) design . (0.00680) correlated according (0.00680) conceptual entities (0.00680) communication is (0.00680) communication are (0.00680) chosen since (0.00680) chosen from (0.00680) choices being (0.00680) certain physical (0.00680) can be (0.00680) being equally (0.00680) be regarded (0.00680) be designed (0.00680) be chosen (0.00680) at the (0.00680) at one (0.00680) at another (0.00680) aspects of (0.00680) aspect is (0.00680) as a (0.00680) are irrelevant (0.00680) are correlated (0.00680) approximately a (0.00680) another point (0.00680) all choices (0.00680) actually be (0.00680) actual message (0.00680) according to (0.00680) a set (0.00680) a message (0.00680) a measure (0.00680) These semantic (0.00680) The system (0.00680) The significant (0.00680) The fundamental (0.00680) If the (0.00680) Frequently the (0.00680) ; that (0.00680) . These (0.00680) . If (0.00680) . Frequently (0.00680) .) (0.00680) , not (0.00680) , all (0.00680)) can (0.00680) (. (0.00680)

Abbildung 2.29: Die empirischen Wahrscheinlichkeiten der Bigram-Datentypen aus Shannons Zitat (absteigend sortiert nach Wahrscheinlichkeiten)

Als Beispiel werden in Abbildungen 2.29 bzw. 2.28 die Bigram- und Trigram-Datentypen zusammen mit ihren empirischen Wahrscheinlichkeiten in Shannons Zitat (siehe Abschnitt 2.1) gezeigt. Shannons Zitat besteht aus 148 Wörtern, also aus 147 Bigram- und 146 Trigram-Datentoken. Nur fünf Bigram-Datentypen kommen in Shannons Zitat zweimal vor, alle anderen, wie auch die Trigram-Datentypen, je einmal.

Es ist immer noch eine sehr schwierige Aufgabe, Trigram- und Bigram-Wahrscheinlichkeiten zu estimieren. Allgemein ist bekannt, dass die Maximum-Likelihood-Estimierung für Trigram- und Bigram-Modelle selbst bei Benutzung sehr grosser Trainingskorpora unbefriedigende Ergebnisse liefert. Ein Grund dafür ist, dass den *unbekannten Daten*, d.h. den Datentypen, die nicht im Trainingskorpus (aber sehr wohl in freiem Text) vorkommen, von der Maximum-Likelihood-Estimierung die bedingte empirische Wahrscheinlichkeit 0 zugewiesen wird. Dies führt bei reinen n-gram-Modellen zu dem unerwünschten Effekt, dass sogar ganze Sätze, wenn sie unbekannte Daten enthalten, die Wahrscheinlichkeit 0 erhalten.

Ein Beispiel soll dieses, für die Sprachmodellierung sehr ernste Problem, verdeutlichen. Der folgende Satz:

$$\textit{the message is unknown}$$

ist im Zusammenhang mit Shannons Zitat sicherlich nicht unplausibel. Seine Wahrscheinlichkeit berechnet sich im Trigram-Modell mit Hilfe von Gleichung 2.5 zu:

$$\begin{aligned} p(\textit{the message is unknown}) &= p(\textit{the message is}) \cdot p(\textit{unknown}|\textit{message is}) \\ &= p(\textit{the message is}) \cdot \frac{p(\textit{message is unknown})}{p(\textit{message is})} \end{aligned}$$

Ein Blick auf Abbildung 2.28 zeigt leider, dass beide massgeblichen Trigram-Datentypen in Shannons Zitat nicht vorkommen. Es sind unbekannte Daten mit der empirischen Wahrscheinlichkeit Null:

$$\begin{aligned} p(\textit{the message is}) &= 0\ , \\ p(\textit{message is unknown}) &= 0\ . \end{aligned}$$

Das Trigram-Modell weist dem angegebenen Satz daher die Wahrscheinlichkeit Null zu. Dies ist sehr unbefriedigend, da der Satz nicht unplausibel erscheint und ein gutes Wahrscheinlichkeits-Modell diesem Satz keine Null-Wahrscheinlichkeit zuweisen sollte. Leider wird sich dieses Ergebnis auch für das Bigram-Modell bestätigen. Dort gilt:

$$\begin{aligned} p(\textit{the message is unknown}) &= p(\textit{the message}) \cdot p(\textit{is}|\textit{message}) \cdot p(\textit{unknown}|\textit{is}) \\ &= p(\textit{the message}) \cdot \frac{p(\textit{message is})}{p(\textit{message})} \cdot \frac{p(\textit{is unknown})}{p(\textit{is})} \end{aligned}$$

[23] Genauso zeigt man allgemeiner, dass durch jedes Sprachmodell eine Wahrscheinlichkeitsverteilung über die Menge aller Sätze mit fester Länge T ist, wenn nur für alle t gilt, dass $\sum_{w_t} p(w_t|h_t) = 1$.

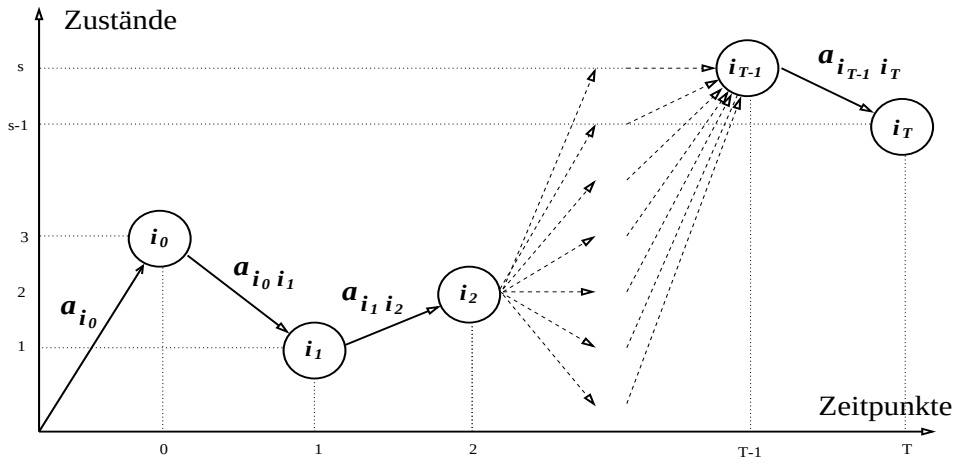


Abbildung 2.30: Markov-Modell

Die empirischen Wahrscheinlichkeiten der massgeblichen Bigramme sind (siehe Abbildung 2.29):

$$\begin{aligned} p(\textit{the message}) &= 0 \ , \\ p(\textit{message is}) &= 0.01360 \ , \\ p(\textit{is unknown}) &= 0.00680 \ . \end{aligned}$$

Auch das Bigram-Modell weist dem Beispielsatz, wegen eines unbekannten Bigram-Datentyps, eine Null-Wahrscheinlichkeit zu. Natürlich kann eingewendet werden, dass Shannons Zitat ein viel zu kleiner Trainingskorpus ist. Dieses Argument ist nicht zu entkräften. Leider muss jedoch festgestellt werden, dass nahezu jeder Trainingskorpus zu klein ist, um reine Trigram-Modelle zu trainieren. Im allgemeinen wird beobachtet, dass für Trigram-Modelle der Anteil unbekannter Daten sehr hoch (ca. 30-40%) ist, selbst wenn einige zehn Millionen Trainingsdaten benutzt werden (Rosenfeld 2000). Es gibt eine Reihe von Techniken, die diese Problematik zu lösen versuchen. Stellvertretend soll die neuere Arbeit von Gao und Lee (2000) genannt werden, in der eine Idee aus dem Information Retrieval benutzt wird, um unbekannten Daten eine Nicht-Null-Wahrscheinlichkeit zuzuweisen. Etwas überraschend, scheint dieses Verfahren die zusätzliche, schöne Eigenschaft zu besitzen, robost bei Änderungen von Stil, Thema und Genre des Trainings-Textes zu reagieren. In Kapitel 4 wird ein Wahrscheinlichkeitsmodell für mehrdimensionale Daten vorgestellt, welches nur selten plausiblen Daten eine Null-Wahrscheinlichkeit zuweist. Eine zukünftige Arbeit könnte daher sein, zu untersuchen, ob dieses Modell ein brauchbares Sprachmodell ergibt.

Die n-gram-Modelle können als spezielle Markov-Modelle aufgefasst werden, welche ihrerseits wieder als spezielle Hidden-Markov-Modelle interpretiert werden können. In einer

Serie von Papieren (Baum und Eagon (1966), Baum und Sell (1968), Baum et al. (1970)) legte eine Gruppe von Mathematikern (Baum, Eagon, Petrie, Sell, Soules, und Weiss) in den 60er Jahren die Grundlage zu einem Trainingsverfahren für Hidden Markov-Modelle. In Baum (1972) findet sich eine sehr elegante Zusammenfassung der wichtigsten Ergebnisse. Im folgenden wird weitgehend die Notation aus diesem Papier verwendet. Hidden-Markov-Modelle basieren mathematisch auf Markov-Modellen. Ein *Markov-Modell* ist ein Wahrscheinlichkeits-Modell, welches durch *Zustände, Startzustands-Wahrscheinlichkeiten* und *Zustandsübergangs-Wahrscheinlichkeiten*[24] charakterisiert wird. Die *Zustände* i sind die Elemente einer endlichen Menge. Da jede endliche Menge durchnummeriert werden kann, ist es üblich, anzunehmen, dass:

$$i \in \{1, \ldots, s\} \; .$$

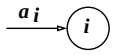
Ein Zustand i wird grafisch durch das folgende Symbol dargestellt:

.

Die *Startzugangs-Wahrscheinlichkeiten* a_i definieren über der Menge der Zustände eine Wahrscheinlichkeitsverteilung:

$$a_i \geq 0, \qquad \sum_{i=1}^{s} a_i = 1, \qquad .$$

Dass ein Zustand i mit Wahrscheinlichkeit a_i ein Startzustand ist, wird grafisch wie folgt dargestellt:

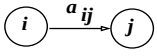


.

Die Zustandsübergangs-Wahrscheinlichkeiten a_{ij} definieren ebenfalls Wahrscheinlichkeitsverteilungen über der Menge der Zustände:

$$a_{ij} \geq 0, \qquad \sum_{j=1}^{s} a_{ij} = 1, \qquad i \in \{1 \ldots s\} \; .$$

Dass ein Zustand j von dem Zustand i mit der Wahrscheinlichkeit a_{ij} erreicht wird, wird symbolisch so festgehalten:



.

Die Abbildung 2.30 zeigt ein Markov-Modell, bei dem nacheinander die $T+1$ Zustände $i_0, i_1, i_2, \ldots, i_{T-1}, i_T$ erreicht werden. Die *Zeitpunkte* $0, 1, 2, \ldots, T-1, T$ werden horizontal abgetragen, und vertikal die erreichten Zustände. In der Abbildung wurden die grafischen Symbole für Zustände, Startzugangs- und Startübergangs-Wahrscheinlichkeiten benutzt.

[24] Transitions-Wahrscheinlichkeiten

Das Markov-Modell weist jeder endlichen Folge $i_0 \dots i_T$ von Zuständen die folgende Wahrscheinlichkeit zu:

$$p(i_0 \dots i_T) \;:=\; a_{i_0} \cdot a_{i_0 i_1} \cdots a_{i_{T-1} i_T} \;=\; a_{i_0} \prod_{t=0}^{T-1} a_{i_t i_{t+1}} \;.$$

Man überzeugt sich leicht, dass durch diese Wahrscheinlichkeiten eine Wahrscheinlichkeitsverteilung auf der Menge $\{1 \dots s\}^{T+1}$ aller Zustandsfolgen der festen Länge $T+1$ gegeben ist:

$$\sum_{i_0 \dots i_T = 1}^{s} p(i_0 \dots i_T) = \sum_{i_0=1}^{s} a_{i_0} \sum_{i_1=1}^{s} a_{i_0 i_1} \dots \sum_{i_T=1}^{s} a_{i_{T-1} i_T} = 1 \;.$$

Die Zustandsfolgen $i_0 \dots i_T$ können für ein Markov-Modell sowohl Ausgabe, als auch Eingabe sein. Wird ein Markov-Modell als Generierungsmodell benutzt, so wird die Zustandsfolge $i_0 \dots i_T$ von dem Markov-Modell mit der Wahrscheinlichkeit $p(i_0 \dots i_T)$ generiert. Hierfür kann der Abbildung 2.30 entnommen werden, dass der Zustand i_t zum Zeitpunkt t generiert wird. Umgekehrt gilt, wenn das Markov-Modell als Analysemodell benutzt wird, dann wird einer eingegebenen Zustandsfolge $i_0 \dots i_T$ die Wahrscheinlichkeit $p(i_0 \dots i_T)$ zugewiesen. Bei Markov-Modellen kann es nicht vorkommen, dass eine Eingabe nicht analysiert werden kann. Natürlich ist dies ein Vorteil von Markov-Modellen gegenüber probabilistischen kontextfreien Grammatiken, da diese im allgemeinen nicht jeden Satz analysieren können. Markov-Modelle sind deshalb robuster als probabilistische kontextfreie Grammatiken[25].

Im folgenden wird gezeigt, dass die n-gram-Modelle als spezielle Markov-Modelle aufgefasst werden können. Es wurde bereits darauf hingewiesen, dass jedes n-gram-Modell durch eine Wahrscheinlichkeitsverteilung $p(hw)$ charakterisiert wird. Hierbei ist h eine beliebige $(n-1)$-dimensionale Geschichte, w ein beliebiges Wort, und hw ein n-gram-Datentyp. Der Schlüsselschritt ist, diese n-gram-Datentypen als Zustände des gesuchten Markov-Modells zu benutzen. Natürlich wird ein solches Markov-Modell eine Folge von n-gram-Datentoken generieren. Leider wird aber die Generierung einer Wortfolge gewünscht (mit derselben Wahrscheinlichkeit wie im gegebenen n-gram-Modell). Werden jedoch die Zustandsübergangs-Wahrscheinlichkeiten des Markov-Modells geschickt gewählt, so kann das Markov-Modell nur solche Folgen von n-gram-Datentoken generieren, die auch bei einer Zerlegung von Text in n-gram-Datentoken entstehen. Das Markov-Modell generiert dann tatsächlich Text (in Gestalt einer Folge von n-gram-Datentoken). Somit stellt sich das Problem so dar, dass die Startzustands- und Zustandsübergangs-Wahrscheinlichkeiten des gesuchten Markov-Modells so angesetzt werden müssen, dass eine Folge von n-gram-

[25] Durch sorgfältige Hinzunahme von speziellen Robustheitsregeln kann man allerdings auch für probabilistische kontextfreie Grammatiken erreichen, dass jede Eingabe eine Analyse erhält (Schulte im Walde und Schmid 2000).

Datentoken vom Markov-Modell dieselbe Wahrscheinlichkeit zugewiesen bekommt, wie ein korrelierender Textkorpus vom n-gram-Modell. Für je zwei Zustände i und j wird deshalb definiert:

$$\begin{aligned} a_j &:= p(hw) \ , \\ a_{ij} &:= \begin{cases} p(w|h) & \text{falls } h = i' \\ 0 & \text{sonst} \end{cases} \end{aligned}$$

In beiden Fällen ist $j = hw$. Die $(n-1)$-dimensionale Wortfolge i' entsteht aus i durch Tilgung des ersten Wortes. Sowohl die a_j als auch die a_{ij} bilden (für festes i) eine Wahrscheinlichkeitsverteilung. Nun zur zweiten Behauptung:

$$\sum_j a_{ij} \ = \ \sum_w p(w|i') \ = \ 1 \ .$$

Für die folgenden n-gram-Datentoken:

$$i_t := w_{t+1} \dots w_{t+n}, \qquad (t = 0 \dots T-n) \ ,$$

folgt unter Benutzung der Gleichung 2.5, dass:

$$\begin{aligned} p(w_1 \dots w_T) &= p(w_1 \dots w_n) \cdot \prod_{t=n+1}^{T} p(w_t|h_t) \\ &= a_{i_0} \cdot \prod_{t=0}^{T-n-1} a_{i_t i_{t+1}} \\ &= p(i_0 \dots i_{T-n}) \ . \end{aligned}$$

Damit ist alles gezeigt: der Text $w_1 \dots w_T$ hat im n-gram-Modell dieselbe Wahrscheinlichkeit wie die Folge $i_0 \dots i_{T-n}$ seiner n-gram-Datentoken im Markov-Modell.

Ein *Hidden-Markov-Modell* ist ein Wahrscheinlichkeits-Modell, welches wie die Markov-Modelle ebenfalls durch Zustände, Startzustands- und Zustandsübergangs-Wahrscheinlichkeiten definiert wird. Im Vergleich zu Markov-Modellen generieren/analysieren Hidden Markov-Modelle aber nicht Zustandsfolgen, sondern Wortfolgen mit Wörtern aus einem vorgegebenen Lexikon. Es wird angenommen, dass die Wörter k des Lexikons ebenfalls durchnummeriert sind, d.h. ohne Einschränkung der Allgemeinheit gilt:

$$k \in \{1 \dots r\} \ .$$

Ein Hidden-Markov-Modell wird im Vergleich zu einem Markov-Modell durch die zusätzliche Angabe der sogenannten *Wortausgabe-Wahrscheinlichkeiten*[26] charakterisiert. Die

[26] Emissions-Wahrscheinlichkeiten

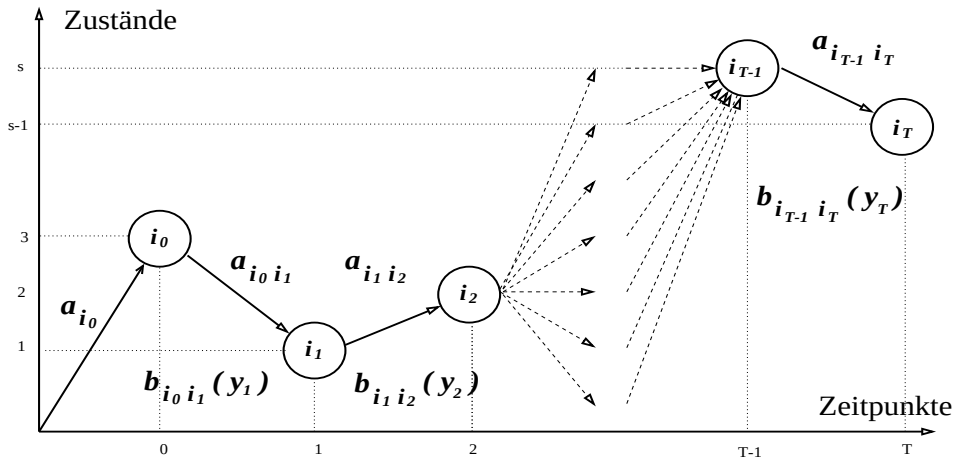
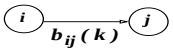


Abbildung 2.31: Hidden-Markov-Modell (HMM)

Wortausgabe-Wahrscheinlichkeiten $b_{ij}(k)$ definieren für je zwei Zustände i, j eine Wahrscheinlichkeitsverteilung über das Lexikon:

$$b_{ij}(k) \geq 0, \qquad \sum_{k=1}^{r} b_{ij}(k) = 1 \qquad i, j \in \{1 \ldots s\} \ .$$

Grafisch wird die Ausgabe eines Wortes k durch die zwei Zustände i, j wie folgt notiert:



.

Die Abbildung 2.31 zeigt ein Hidden-Markov-Modell, bei dem die Zustände

$$i_0, i_1, i_2, \ldots, i_{T-1}, i_T$$

erreicht werden, und die Wortfolge

$$y_1 y_2 \ldots y_T$$

ausgegeben wird. Offensichtlich muss zur Ausgabe einer Wortfolge der Länge T eine Zustandsfolge der Länge $T+1$ durchlaufen werden. Die *Zeitpunkte* $0, 1, 2, \ldots, T-1, T$ werden horizontal abgetragen, vertikal die erreichten Zustände. Die Abbildung 2.31 ergänzt die Abbildung 2.30 genau um die Wortausgabe-Wahrscheinlichkeiten. Der Abbildung entnimmt man, dass das Wort y_t beim Übergang von Zustand i_{t-1} zum Zustand i_t, also zwischen den Zeitpunkten $t-1$ und t generiert wird[27].

Die Wahrscheinlichkeit einer Wortfolge $y_1 \ldots y_T$ ist im Hidden-Markov-Modell, wie folgt definiert:

$$p(y_1 \ldots y_T) \ := \sum_{i_0 \ldots i_T = 1}^{s} p(i_0 \ldots i_T) \cdot p(y_1 \ldots y_T \mid i_0 \ldots i_T) \ .$$

[27] Alternativ (üblicherweise für Tagging und in der Spracherkennung) können die Ausgabewahrschscheinlichkeiten statt an den Übergängen auch an den Zuständen festgemacht werden.

Hierbei ist $p(i_0 \dots i_T)$ die Wahrscheinlichkeit der Zustandsfolge $i_0 \dots i_T$, die bereits für Markov-Modelle definiert wurde. Neu ist hingegen die *bedingte Wahrscheinlichkeit der Wortfolge $y_1 \dots y_T$, gegeben der Zustandsfolge $i_0 \dots i_T$*. Diese ist wie folgt definiert:

$$p(y_1 \dots y_T \mid i_0 \dots i_T) := b_{i_0 i_1}(y_1) \cdots b_{i_{T-1} i_T}(y_T) = \prod_{1 \le t \le T} b_{i_{t-1} i_t}(y_t) \ .$$

Zusammenfassend:

$$p(y_1 \dots y_T) \ = \ \sum_{i_0 \dots i_T = 1}^{s} a_{i_0} \cdot a_{i_0 i_1} \cdot b_{i_0 i_1}(y_1) \cdots a_{i_{T-1} i_T} \cdot b_{i_{T-1} i_T}(y_T) \ .$$

Es ist leicht einzusehen, dass die bedingten Wahrscheinlichkeiten $p(y_1 \dots y_T \mid i_0 \dots i_T)$ eine Wahrscheinlichkeitsverteilung auf der Menge $\{1 \dots r\}^T$ aller Wortfolgen liefern:

$$\sum_{y_1 \dots y_T = 1}^{r} p(y_1 \dots y_T \mid i_0 \dots i_T) = \sum_{y_1 = 1}^{r} b_{i_0 i_1}(y_1) \cdots \sum_{y_T = 1}^{r} b_{i_{T-1} i_T}(y_T) = 1 \ .$$

Hieraus folgt, dass das Hidden-Markov-Modell ebenfalls eine Wahrscheinlichkeitsverteilung auf der Menge aller Wortfolgen einer festen Länge T liefert:

$$\begin{aligned}
\sum_{y_1 \dots y_T = 1}^{r} p(y_1 \dots y_T) &= \sum_{y_1 \dots y_T = 1}^{r} \sum_{i_0 \dots i_T = 1}^{s} p(i_0 \dots i_T) \cdot p(y_1 \dots y_T \mid i_0 \dots i_T) \\
&= \sum_{i_0 \dots i_T = 1}^{s} p(i_0 \dots i_T) \cdot \sum_{y_1 \dots y_T = 1}^{r} p(y_1 \dots y_T \mid i_0 \dots i_T) \\
&= \sum_{i_0 \dots i_T = 1}^{s} p(i_0 \dots i_T) \cdot 1 \\
&= 1 \ .
\end{aligned}$$

Das folgende Theorem, in dem ein effizienter Trainingsalgorithmus gegeben wird, begründet die Popularität der Hidden Markov-Modelle.

Theorem (Forward-Backward-Algorithmus, (Baum 1972)): Für ein gegebenes Hidden Markov Modell $p(.)$ und eine Wortfolge $y_1 \dots y_T$ von *Trainingsdaten*, definieren die folgenden Zustandsübergangs- und Wortausgabe-Wahrscheinlichkeiten:

$$\hat{a}_{ij} \ := \ \frac{a_{ij} \cdot \frac{\partial \ p(y_1 \dots y_T)}{\partial \ a_{ij}}}{\sum_j a_{ij} \cdot \frac{\partial \ p(y_1 \dots y_T)}{\partial \ a_{ij}}} \ ,$$

$$\hat{b}_{ij}(k) \ := \ \frac{b_{ij}(k) \cdot \frac{\partial \ p(y_1 \dots y_T)}{\partial \ b_{ij}}}{\sum_k b_{ij}(k) \cdot \frac{\partial \ p(y_1 \dots y_T)}{\partial \ b_{ij}(k)}} \ ,$$

ein *re-estimiertes Hidden Markov-Modell* $\hat{p}(.)$ mit:

$$\hat{p}(y_1 \dots y_T) \ge p(y_1 \dots y_T) \ .$$

Gleichheit tritt genau dann ein, wenn (für alle Zustände i, j und alle Wörter k) gilt, dass:

$$\hat{a}_{ij} = a_{ij}, \qquad \hat{b}_{ij}(k) = b_{ij}(k) \ .$$

Als Funktion ihrer Wahrscheinlichkeitsparameter betrachtet, liegt dann an dieser Stelle ein Extremwert der Funktion $p(y_1 \dots y_T)$ vor. Für die *re-estimierten Zustandsübergangs- und Wortausgabe-Wahrscheinlichkeiten* gilt zusätzlich, dass sie sich effizient berechnen lassen:

$$\hat{a}_{ij} := \frac{\sum_t \alpha_t(i) \cdot \beta_{t+1}(j) \cdot a_{ij} \cdot b_{ij}(y_{t+1})}{\sum_t \alpha_t(i) \cdot \beta_t(i)} \ ,$$

$$\hat{b}_{ij}(k) := \frac{\sum_{y_t=k} \alpha_t(i) \cdot \beta_{t+1}(j) \cdot a_{ij} \cdot b_{ij}(y_{t+1})}{\sum_t \alpha_t(i) \cdot \beta_{t+1}(j) \cdot a_{ij} \cdot b_{ij}(y_{t+1})} \ .$$

Hierbei sind:

$$\alpha_{t+1}(j) := \sum_{i=1}^{s} \alpha_t(i) \cdot a_{ij} \cdot b_{ij}(y_{t+1}) \qquad j = 1 \dots s, \quad t = 0 \dots T-1,$$

$$\beta_t(i) := \sum_{j=1}^{s} \beta_{t+1}(j) \cdot a_{ij} \cdot b_{ij}(y_{t+1}) \qquad i = 1 \dots s, \quad t = T-1 \dots 0,$$

die sogenannten *Forward-* und *Backward-Variablen.*

Bemerkung: Die *Wahrscheinlichkeit des Trainingstextes*:

$$p(y_1 \dots y_T) = \sum_{i=1}^{s} \alpha_t(i) \cdot \beta_t(i) \ ,$$

die konstant in t ist, kann mit $4s^2 \cdot T$ Multiplikationen berechnet werden. Wenn anstatt der Forward- und Backward-Variablen die Definition von $p(y_1 \dots y_T)$ benutzt wird, werden $2T \cdot s^{T+1}$ Multiplikationen benötigt. Natürlich übertragen sich analoge Aussagen auch auf die partiellen Ableitungen von $p(y_1 \dots y_T)$. Dies führt dazu, dass der gesamte Forward-Backward-Algorithmus sehr effizient ist. Ferner ist interessant, dass Baum in seiner Arbeit auch die wenig bekannten Re-estimierungsformeln für Hidden-Markov-Modelle mit einem nicht-diskreten Lexikon angibt. In Kapitel 3 werden Baums Formeln im Rahmen des EM-Algorithmus für probabilistische kontextfreie Grammatiken als einfache Spezialfälle[28] bewiesen werden.

Natürlich sind Hidden-Markov-Modelle sehr viel allgemeinere Wahrscheinlichkeitsmodelle als Markov-Modelle. Im folgenden soll dies gezeigt werden. Um ein Markov-Modell mit

[28] Sehr interessant ist, dass hierbei ein Trainingskorpus benutzt werden muss, der aus einem einzigen Satz besteht. Ausserdem wird die probabilistische kontextfreie Grammatik *gekoppelte Grammatikregel--Wahrscheinlichkeiten* besitzen, d.h. Grammatikregel-Wahrscheinlichkeiten, welche, neben der Standard-Nebenbedingung, weiteren Nebenbedingungen genügen müssen.

$S \to X_i^0$	(a_i)	für alle Zustände i
$X_i^t \to E_{ij}\ X_j^{t+1}$	(a_{ij})	für alle Zustände i, j und Zeiten $t \in \{0 \ldots T-1\}$
$X_i^T \to \epsilon$	(1)	für alle Zustände i
$E_{ij} \to k$	$(b_{ij}(k))$	für alle Zustände i, j und Wörter k

($T \in \mathcal{N}$ fest gewählt)

Abbildung 2.32: Hidden Markov-Modell als probabilistische kontextfreie Grammatik

einem Hidden-Markov-Modell nachzubilden, muss das Lexikon des Hidden-Markov-Modells mit der Menge der Zustände des vorgegebenen Markov-Modells identifiziert werden. D.h.

$$\{1 \ldots r\} := \{1 \ldots s\} \ .$$

Ferner muss das Hidden-Markov-Modell beim Übergang von Zustand i nach Zustand j deterministisch den Zustand i ausgeben:

$$b_{ij}(k) := \begin{cases} 1 & \text{für } k = i, \\ 0 & \text{sonst} \end{cases}$$

In dem so definierten Hidden-Markov-Modell gilt dann:

$$\begin{aligned} p(y_1 \ldots y_T \mid i_0 \ldots i_T) &= b_{i_0 i_1}(y_1) \cdots b_{i_{T-1} i_T}(y_T) \\ &= \begin{cases} 1 & \text{für } y_1 \ldots y_T = i_0 \ldots i_{T-1}, \\ 0 & \text{sonst} \end{cases} \end{aligned}$$

Folglich:

$$\begin{aligned} p(y_1 \ldots y_T) &= \sum_{i_0 \ldots i_T = 1}^{s} p(i_0 \ldots i_T) \cdot p(y_1 \ldots y_T \mid i_0 \ldots i_T) \\ &= \sum_{i_T = 1}^{s} p(i_0 \ldots i_T) \\ &= p(i_0 \ldots i_{T-1}) \ . \end{aligned}$$

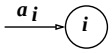
D.h. die Wahrscheinlichkeit der Wortfolge $y_1 \ldots y_T$ im Hidden Markov-Modell korreliert mit der Wahrscheinlichkeit der Zustandsfolge $i_0 \ldots i_{T-1}$ im vorgegebenen Markov-Modell.

Sehr viel interessanter als der Zusammenhang zwischen Markov- und Hidden Markov-Modellen ist der Zusammenhang zwischen Hidden Markov-Modellen und probabilistischen kontextfreien Grammatiken. Die Abbildung 2.32 zeigt eine probabilistische kontextfreie Grammatik, die ein vorgegebenes Hidden Markov-Modell nachbildet. In der ersten Spalte

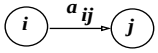
sind die kontextfreien Grammatikregeln angegeben, in der zweiten Spalte die Grammatikregel-Wahrscheinlichkeiten. In der Grammatik kommen die folgenden Grammatikkategorien vor:

$$S,\ X_i^t, E_{ij} \qquad (i,j \in \{1 \ldots s\}, \quad t \in \{0, \ldots T\}) \ .$$

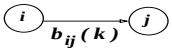
Die Grammatikkategorie X_i^t korreliert mit dem Zustand i zum Zeitpunkt t, die Grammatikkategorie E_{ij} mit einem Zustandsübergang von i nach j (zu einem beliebigen Zeitpunkt). Somit wird der Startzustand



durch die Grammatikregel $S \rightarrow X_i^0\ (a_i)$, der Zustandsübergang



durch die Grammatikregel $X_i^t \rightarrow E_{ij}\ X_j^{t+1}\ (a_{ij})$, sowie die Wortausgabe



durch die Grammatikregel $E_{ij} \rightarrow k\ (b_{ij}(k))$ ausgedrückt. Bemerkenswert ist, dass die Zeitpunkte

$$t \in \{0, \ldots T\}$$

in der Darstellung eines Hidden Markov-Modells als probabilistische kontextfreie Grammatik unverzichtbar sind[29], um die Stationarität (oder Zeit-Invarianz) des Hidden Markov-Modells in den Grammatikregel-Wahrscheinlichkeiten explizit ausdrücken zu können ($i,j \in \{1 \ldots s\}$):

$$p(X_i^0 \rightarrow E_{ij}\ X_j^1) \ = \ \ldots \ = \ p(X_r^{T-1} \rightarrow E_{ij}\ X_j^T) \ = \ a_{ij} \ .$$

Es ist sehr leicht nachzuprüfen, dass die angegebene Grammatik die Standard-Nebenbedingung (Booth und Thompson 1973) für probabilistische kontextfreie Grammatiken erfüllt.

Im folgenden wird gezeigt, dass die probabilistische kontextfreie Grammatik aus Abbildung 2.32 die Sprache $\{1 \ldots r\}^T$ besitzt und jeder Wortfolge aus ihrer Sprache dieselbe

[29] Die folgende probabilistische kontextfreie Grammatik

$S \rightarrow X_i$	(a_i)	für alle Zustände i
$X_i \rightarrow E_{ij}\ X_j$	(a_{ij})	für alle Zustände i,j
$X_i \rightarrow \epsilon$	(1)	für alle Zustände i
$E_{ij} \rightarrow k$	$(b_{ij}(k))$	für alle Zustände i,j und Wörter k

hat zwei Mängel: Erstens, sie ist rekursiv und erzeugt die Sprache $\cup_{t=0}^{\infty}\{1 \ldots r\}^t$ (gesucht ist aber eine Grammatik, die nur die Wortfolgen der Länge T generiert). Zweitens, sie verletzt die Standard-Nebenbedingung für probabilistische kontextfreie Grammatiken: die Wahrscheinlichkeiten der Regeln mit der linken Seite X_i summieren sich zu der Zahl 2 auf.

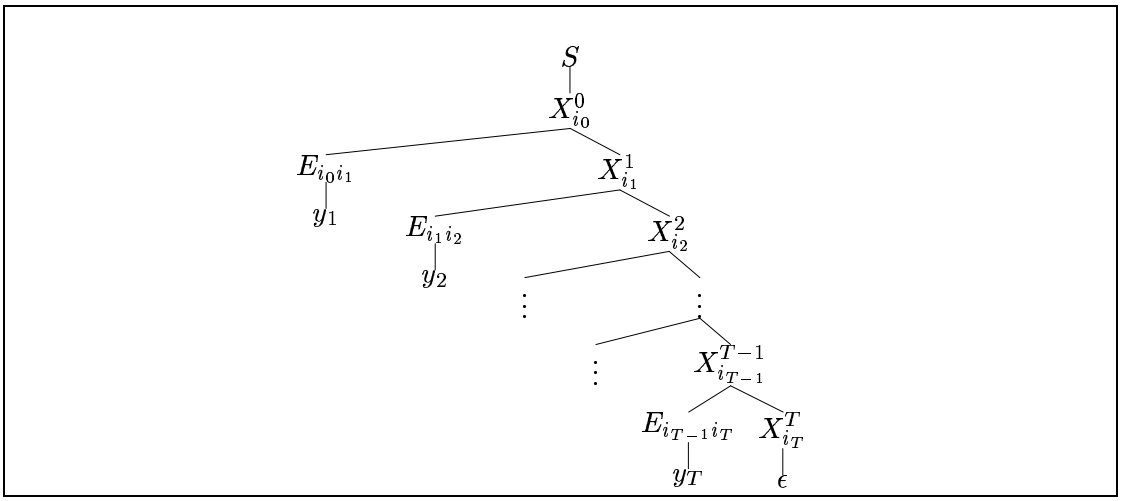


Abbildung 2.33: Syntaxanalyse mit einem Hidden Markov-Modell

Wahrscheinlichkeit zuweist, wie das Hidden Markov-Modell, aus dem sie konstruiert ist. Abbildung 2.33 zeigt für die Wortfolge $y_1 \dots y_T$ einen möglichen Syntaxbaum $x_{i_0 \dots i_T}$. Offensichtlich sorgt die Grammatikregel $X_r^T \to \epsilon$ (1) dafür, dass nur Wortfolgen mit der Länge T analysiert werden können. Die Sprache der abgebildeten Grammatik ist also $\{1 \dots r\}^T$. Die Wahrscheinlichkeit des Syntaxbaums $x_{i_0 \dots i_T}$ ist das Produkt aller seiner Grammatikregel-Wahrscheinlichkeiten. Es gilt:

$$p(x_{i_0 \dots i_T}) \;=\; a_{i_0} \cdot a_{i_0 i_1} \cdot b_{i_0 i_1}(y_1) \cdots a_{i_{T-1} i_T} \cdot b_{i_{T-1} i_T}(y_T) \;=\; p(y_1 \dots y_T \mid i_0 \dots i_T) \;.$$

Die Wahrscheinlichkeit dieses Syntaxbaums ist also gleich der bedingten Wahrscheinlichkeit von Wort- gegeben Zustandsfolge im Hidden-Markov-Modell.

Die Menge aller Syntaxbäume der Wortfolge $y_1 \dots y_T$ ist:

$$\mathcal{T}(y_1 \dots y_T) \;=\; \{x_{i_0 \dots i_T} \mid i_0, \dots, i_T \in \{1 \dots s\}\}$$

Somit gilt für die probabilistische kontextfreie Grammatik aus Abbildung 2.32, dass:

$$\begin{aligned} p(y_1 \dots y_T) &= p\left(\mathcal{T}(y_1 \dots y_T)\right) \\ &= \sum_{i_0 \dots i_T} p(x_{i_0 \dots i_T}) \\ &= \sum_{i_0 \dots i_T} p(i_0 \dots i_T) \cdot p(y_1 \dots y_T \mid i_0 \dots i_T) \;. \end{aligned}$$

Die Wahrscheinlichkeit der Wortfolge $y_1 \dots y_T$ in der probabilistischen kontextfreien Grammatik stimmt daher mit ihrer Wahrscheinlichkeit im Hidden Markov-Modell überein.

Damit ist das angekündigte Ziel dieses Abschnitts erreicht, zu zeigen, dass die Hidden Markov-Modelle als spezielle, sehr einfache probabilistische kontextfreie Gramma-

$S \to X_i^0$	(a_i)	für alle Zustände i
$X_i^t \to E_{ij}\ X_j^{t+1}$	(a_{ij})	für alle Zustände i, j und Zeiten $t \in \{0 \ldots T-1\}$
$X_i^T \to \epsilon$	(1)	für alle Zustände i
$E_{ij} \to i$	(1)	für alle Zustände i, j

($T \in \mathcal{N}$ fest gewählt)

Abbildung 2.34: Markov-Modell als probabilistische kontextfreie Grammatik

$S \to X_{hw}^0$	$p(hw)$	für alle n-gram-Datentypen hw
$X_{vh}^t \to E_{vh,hw}\ X_{hw}^{t+1}$	$p(w\|h)$	für alle Wörter v, w und Geschichten h
$X_{hw}^T \to \epsilon$	(1)	für alle n-gram-Datentypen hw
$E_{vh,hw} \to vh$	(1)	für alle n-gram-Datentypen vh und hw

($T \geq n$ fest gewählt)

Abbildung 2.35: n-gram-Modell als probabilistische kontextfreie Grammatik

tiken angesehen werden können. In Kapitel 3 wird gezeigt werden, dass Baums Reestimierungsformeln einfache Spezialfälle des EM-Algorithmus für probabilistische kontextfreie Grammatiken sind.

Der Vollständigkeit halber, werden in den Abbildungen 2.34 und 2.35 die probabilistischen kontextfreien Grammatiken angegeben, die ein vorgegebenes Markov- bzw. n-gram-Modell simulieren. Natürlich gibt es einfachere Grammatiken, die dasselbe leisten. Die angegebenen Grammatiken haben jedoch den Vorteil, den vorgestellten Argumenten möglichst getreu zu folgen, dass n-gram-Modelle Spezialfälle der Markov-Modelle sind, und diese wiederum Spezialfälle der Hidden Markov-Modelle. Damit soll dieser Abschnitt beschlossen werden.

Kapitel 3

Der EM-Algorithmus

Als Schwerpunkt des vorangegangenen Kapitels wurden die wichtigsten stochastischen Modelle der maschinellen Sprachverarbeitung mit ihren spezifischen Anwendungen und Trainingsverfahren vorgestellt und gezeigt, dass jedes dieser Modelle als eine *probabilistische kontextfreie Grammatik* aufgefasst werden kann. In diesem Kapitel soll diese Arbeit ergänzt werden, indem formal bewiesen wird, dass das Trainingsverfahren der probabilistischen kontextfreien Grammatiken, der bekannte *Inside-Outside-Algorithmus* (Baker (1979), Lari und Young (1990)), eine *dynamische Programmiervariante* einer speziellen Instanz des *EM-Algorithmus* (Dempster et al. 1977) ist. Unter Benutzung der Ergebnisse aus Kapitel 2 führt dieses Ergebnis zu einer ersten Folgerung:

- Die wichtigsten stochastischen Modelle der maschinellen Sprachverarbeitung, das *n-gram Modell*, das *Markov-Modell*, das *Hidden-Markov-Modell*, die *Baumbank-Grammatik* und die *probabilistische kontextfreie Grammatik* können mit speziellen EM-Algorithmen trainiert werden.

In den nachfolgenden Kapiteln wird gezeigt werden, dass sich zu diesen Modellen noch ein *Klassifizierungsverfahren für mehrdimensionale linguistische Datentypen* und die *lexikalisierten kontextfreien Grammatiken*, sowie die auf *log-linearen Modellen* basierenden *stochastischen unifikationsbasierten Grammatiken* gesellen werden. "Die Welt" ist daher, in Abwandlung eines Zitats, kein *endlicher Automat*, sondern ein EM-Algorithmus.

Eine zweite Konsequenz betrifft den Inside-Outside-Algorithmus selbst. In der Literatur wird oft behauptet, dass Baker (1979) oder Lari und Young (1990) bewiesen hätten, dass die mit dem Inside-Outside-Algorithmus iterativ trainierten probabilistischen Grammatiken *mononton wachsende Log-Likelihood-Werte* (bzw. Korpus-Wahrscheinlichkeiten) aufweisen, und gegen ein *lokales Maximum der Log-Likelihood-Funktion* konvergieren. Richtig ist jedoch, dass Baker (1979) lediglich eine intuitive Verallgemeinerung des *Forward-*

Backward-Algorithmus (Baum 1972) vorgenommen hat und Baum (1972) den formalen "Monotonie- und Konvergenzbeweis" explizit für Hidden-Markov-Modelle angab. Leider kommt hinzu, dass Baker (1979) nur Trainingskorpora betrachtete, die aus einem einzigen Satz bestehen. Lari und Young (1990) beseitigten diese Schwäche und verallgemeinerten den Inside-Outside-Algorithmus, um ihn auf beliebige Trainingskorpora anwenden zu können. Auch in dieser Arbeit fehlt aber ein formaler Monotonie- und Konvergenzbeweis. Es ist interessant, dass Lari und Young (1991) darauf hinweisen, dass gewisse im Inside-Outside-Algorithmus benutzte *Grammatikregel-Counts* als *erwartete Grammatikregelhäufigkeiten* interpretiert werden können. Leider wurden die mathematischen Grundlagen der EM-Theorie (Dempster et al. 1977) nicht angewendet, um diese *intuitiv erfasste Schlüsselbeobachtung* in dieser Arbeit auch *formal zu beweisen.*

- Nach bestem Wissen des Verfassers wird daher die allgemein bekannte Ansicht, dass der Inside-Outside-Algorithmus eine dynamische Programmiervariante des EM-Algorithmus ist, zum ersten Mal in der vorliegenden Arbeit formal bewiesen.

Als *dynamische Programmiervariante des EM-Algorithmus* erbt der Inside-Outside-Algorithmus dessen Konvergenzeigenschaften. Diese stimmen allerdings nicht völlig mit den Eigenschaften überein, die man gemeinhin dem Inside-Outside-Algorithmus zuschreibt, und stellen somit ein weiteres wichtiges Ergebnis dieses Kapitels dar:

- Die mit dem Inside-Outside-Algorithmus iterativ trainierten probabilistischen Grammatiken weisen *mononton wachsende Log-Likelihood-Werte* (bzw. Korpus-Wahrscheinlichkeiten) auf, und ein Grenzwert dieser Folge ist ein *stationärer Wert der Log-Likelihood-Funktion* und in gewissen Fällen auch ein *lokales Maximum.*

Eine kurze Darstellung dieser Ergebnisse findet sich in Prescher (2001b), und etwas ausf ührlicher in Prescher (2001c).

Das Hauptziel dieses Kapitels besteht darin, die formalen Grundlagen des EM-Algorithmus vorzustellen und zu zeigen, wie er in der Linguistik nutzbringend angewendet werden kann. Dazu gehört, eine Interpretation des EM-Algorithmus anzugeben und den EM-Algorithmus in einen grösseren Kontext zu stellen. In diesem Kapitel gelingt dies, indem der EM-Algorithmus als eine besonders gut motivierbare Instanz eines neuen *Meta-Algorithmus* vorgestellt wird.

- Der *stochastische Bootstrap-Algorithmus (SB-Algorithmus)* wurde vom Verfasser konzipiert, um ein beliebiges stochastisches Trainingsverfahren, das mit *annotierten Datentypen* arbeitet, mit Hilfe einer *symbolischen Analyse-Komponente* in ein

iteratives stochastisches Trainingsverfahren zu überführen, welches lediglich einen *Trainingskorpus von unannotierten Datentypen* benötigt.

Der EM-Algorithmus ist eine Instanz des SB-Algorithmus, die man erhält, indem im SB-Algorithmus als zugrundeliegendes Trainingsverfahren die *Maximum-Likelihood-Estimierung* für annotierte Datentypen benutzt wird. Leider wird dem EM-Algorithmus in der stochastischen Linguistik immer noch mit Skepsis begegnet, da in einigen bekannteren Arbeiten relativ schlechte Ergebnisse reportiert wurden, die mit dem EM-Algorithmus erzielt wurden. Nach einem schärferen Blick auf viele dieser Arbeiten kann aber vermutet werden, dass die schlechten Ergebnisse eher dem Experimentator als dem EM-Algorithmus zuzuschreiben sind. Ein besonderes Anliegen dieses Kapitels ist es daher, das Training mit dem EM-Algorithmus und die Evaluierung der erhaltenen stochastischen Modelle so zu beschreiben, dass jeder in die Lage versetzt wird, mit dem EM-Algorithmus möglichst bessere Ergebnisse zu erzielen, als mit alternativen und teuren Trainingsverfahren, die auf manuell erstellten Trainingskorpora basieren.

- Die freien Parameter des EM-Algorithmus (*Grösse des Trainingskorpus unvollständiger Datentypen*, *Anzahl Trainingsiterationen* und *Startparameter*) werden vorgestellt und es wird vorgeschlagen, wie ein gegebener *Korpus von unvollständigen Datentypen* dazu benutzt werden kann, diese Parameter optimal einzustellen und systematisch zu evaluieren.

Für eine nutzbringende Anwendung des EM-Algorithmus ist es nicht nur vorteilhaft, zu verstehen, wie der EM-Algorithmus arbeitet, sondern auch zu wissen, welche formalen Eigenschaften der EM-Algorithmus besitzt. Hierbei ist interessant, dass von den vier Theoremen, die Dempster et al. (1977) bei der Einführung des EM-Algorithmus angaben, nur zwei Theoreme korrekt waren, nämlich die Theoreme zur *wachsenden Monotonie* und zur *Konvergenz (gegen stationäre Werte)* der Log-Likelihood. Wu (1983) ergänzte diese Aussagen um Aussagen zur *Konvergenz gegen lokale Maxima*, sowie zur *Parameter-Konvergenz der Log-Likelihood*. Der Verfasser stellt diese Ergebnisse vor und ergänzt sie um einige eigene einfache Aussagen. Daran anschliessend werden alle vorgestellten Techniken und Ergebnisse dieses Kapitels auf die *probabilistischen kontextfreien Grammatiken* angewendet, um den EM-Algorithmus für kontextfreie Grammatiken und seine Eigenschaften vorzustellen. Dieses Kapitel gliedert sich in die folgenden Abschnitte:

Im Abschnitt 3.1 wird der *Inside-Outside-Algorithmus* vorgestellt, mit dem es möglich ist, eine gegebene kontextfreie Grammatik auf unannotiertem Text (d.h. ohne Hilfe einer manuell erstellten Baumbank) iterativ zu trainieren. Die Darstellung dieses Abschnitts folgt in seinen wichtigsten Teilen der Darstellung von Lari und Young (1990). Zunächst werden die

Outside-Parsewälder und -Wahrscheinlichkeiten vorgestellt, die in einem gewissen Sinn die Gegenstücke der in Kapitel 2 eingeführten *Inside-Parsewälder und -Wahrscheinlichkeiten* sind. Nach Vorstellung der *Rekursionsformeln* für die Inside- und Outside-Parsewälder und Wahrscheinlichkeiten wird der *Outside-Algorithmus* vorgestellt, um explizit zu demonstrieren, dass auch die Outside-Wahrscheinlichkeiten sehr effizient berechnet werden können. Hierbei ist bemerkenswert, dass der Outside-Algorithmus, im Gegensatz zum Inside-Algorithmus, leider keine Instanz des Semiring-Parsing-Algorithmus ist (siehe Abschnitt 2.7). Anschliessend werden die Re-Estimierungsformeln des Inside-Outside-Algorithmus für einen Trainingskorpus, bestehend aus einem einzigen Satz (Baker 1979) und für beliebig grosse Trainingskorpora (Lari und Young 1990) vorgestellt. Der Schlüsselschritt ist die Benutzung sogenannter *Grammatikregel- und Kategorie-Counts*, die auf den Inside- und Outside-Wahrscheinlichkeiten basieren und dem Algorithmus seinen Namen geben. Die *Grammatikregel- und Kategorie-Counts* werden für einen gegebenen Trainingskorpus von Sätzen satzweise berechnet, sowie über dem gesamten Trainingskorpus akkumuliert, um daraus re-estimierte Grammatikregel-Wahrscheinlichkeiten zu berechnen. Fälschlicherweise wird in der Literatur oft behauptet, dass Baker (1979) oder Lari und Young (1990) bewiesen hätten, dass die mit dem Inside-Outside-Algorithmus iterativ trainierten probabilistischen Grammatiken mononton wachsende Log-Likelihoodwerte (bzw. Korpus-Wahrscheinlichkeiten) aufweisen, und gegen ein lokales Maximum der Log-Likelihood-Funktion konvergieren (was tatsächlich nicht immer der Fall ist). Zum Abschluss des Abschnitts 3.1 wird darauf hingewiesen, dass die mit dem Inside-Outside-Algorithmus trainierten probabilistischen Grammatiken in der Regel *linguistisch evaluiert* werden sollten, und es wird vorgeschlagen, wie die freien Parameter des Inside-Outside-Algorithmus (Grösse des Trainingskorpus, Iterationsanzahl, Startgrammatiken) in einer *systematischen linguistischen Evaluierung* optimal festgelegt werden können.

In Abschnitt 3.2 wird der EM-Algorithmus zunächst formal anhand seiner symbolischen und stochastischen Komponenten eingeführt. Hierzu werden die acht Basisbegriffe: *unvollständiger Datentyp*, *empirische Wahrscheinlichkeitsverteilung der unvollständigen Datentypen*, *vollständiger Datentyp*, *symbolische Analyse-Komponente*, *Parameterraum*, *Startparameter*, *parametrisierte Wahrscheinlichkeitsverteilung der vollständigen Datentypen*, sowie der Begriff der *EM-Parameter-Abbildung* eingeführt, sowie die fünf folgenden Begriffe aus den Basisbegriffen abgeleitet: *parametrisierte Wahrscheinlichkeitsverteilung der unvollständigen Datentypen*, *parametrisierte bedingte Wahrscheinlichkeit eines vollständigen gegeben einem unvollständigen Datentyp*, *EM-Hilfsfunktion*, *Erwartungswert-Schritt (E-Schritt)* und *Maximierungs-Schritt (M-Schritt)*. Daran anschliessend wird der EM-Algorithmus aus prozeduraler Sicht dargestellt. Hierbei wird eine Version des EM-Algorithmus mit expliziter Variation der Startparameter vorgestellt, da diesem freien Parameter des

EM-Algorithmus in der Regel die grösste Bedeutung zukommt. Es wird darauf hingewiesen, dass der EM-Algorithmus eine Sequenz von Modellen trainiert, deren Log-Likelihoodwerte monoton wachsend sind und unter zusätzlichen Bedingungen gegen ein lokales Maximum konvergieren. Zusätzlich gilt in diesem Fall, dass auch die Parameter der trainierten Modelle konvergieren. Es wird allerdings bemerkt, dass es trotz dieser ausgezeichneten theoretischen Eigenschaften sinnvoll ist, eine systematische linguistische Evaluierung aller trainierten Modelle vorzunehmen.

In Abschnitt 3.3 werden zwei Ziele verfolgt: (i) um die Anwendung des EM-Algorithmus zu erleichtern, werden seine wichtigsten Komponenten erneut vorgestellt, wobei zunächst die symbolischen und dann die stochastischen Komponenten behandelt werden, (ii) es werden zwei alternative Implementierungen des EM-Algorithmus vorgestellt, die zwar eine unveränderte Funktionalität aufweisen, aber tiefere Einblicke in seine Funktionsweise ermöglichen. Bei der Interpretation des EM-Algorithmus wird die Einführung der *empirischen Wahrscheinlichkeitsverteilung der vollständigen Datentypen* eine Schlüsselrolle spielen. Mit ihrer Hilfe kann gezeigt werden, dass die EM-Hilfsfunktion eine *Log-Likelihood auf einem Korpus von vollständigen Datentypen* ist, was dazu führt, dass der EM-Algorithmus als *iterative Maximum-Likelihood-Estimierung vollständiger Datentypen* gedeutet werden kann und damit eine besonders gut motivierte Instanz des in Abschnitt 3.4 vorgestellten SB-Algorithmus ist.

In Abschnitt 3.4 wird der EM-Algorithmus in den Kontext des hier neu eingeführten *stochastischen Bootstrap-Algorithmus (SB-Algorithmus)* gestellt. Der SB-Algorithmus wurde konzipiert, um ein beliebiges *stochastisches Trainingsverfahren auf vollständigen Datentypen* in ein *iteratives stochastisches Trainingsverfahren auf unvollständigen Datentypen* zu transformieren. Als Schlüsselschritt des SB-Algorithmus fungiert erneut die *empirische Wahrscheinlichkeitsverteilung der vollständigen Datentypen.* Der SB-Algorithmus nutzt die simple Idee aus, dass diese Wahrscheinlichkeitsverteilung mit jedem vorgegebenen Trainingsverfahren für vollständige Datentypen kombinierbar ist. Der Vorteil des SB-Algorithmus gegenüber der direkten Benutzung des zugrundeliegenden überwachten Trainingsverfahrens liegt darin, dass es mit dem SB-Algorithmus unproblematisch ist, *beliebig viele Modellvarianten* (Startparameter, Anzahl Iterationen) auf *ausreichend grossen Trainingskorpora* zu trainieren. Der Nachteil des SB-Algorithmus liegt darin, dass er eine (zumindest einfache) *symbolische Analysekomponente* voraussetzt. Ausserdem muss von Instanz zu Instanz geklärt werden, ob der SB-Algorithmus die guten und schlechten Eigenschaften des zugrundeliegenden stochastischen Trainingsverfahrens für vollständige Datentypen erbt. In jedem Fall gilt aber, dass der EM-Algorithmus eine besonders gut motivierbare Instanz des SB-Algorithmus ist, die man erhält, indem im SB-Algorithmus als zugrundeliegendes

überwachtes Trainingsverfahren die Maximum-Likelihood-Estimierung auf vollständigen Datentypen benutzt wird.

Abschnitt 3.5 ist dem praktischen Training und der Evaluierung des EM-Algorithmus gewidmet. Es werden die freien Parameter des EM-Algorithmus vorgestellt (*Grösse des Trainingskorpus unvollständiger Datentypen*, *Anzahl Trainingsiterationen* und *Startparameter*) und vorgeschlagen, wie ein gegebener Korpus von unvollständigen Datentypen dazu benutzt werden kann, diese Parameter optimal einzustellen und zu evaluieren. Kern des Vorschlags ist eine zweimalige *Aufteilung* und teilweise *Annotation* des gegebenen Korpus, die in drei Korpora resultiert: (i) einen *Trainingskorpus (unvollständiger Datentypen)*, (ii) einen *Entwicklungs-Testkorpus (vollständiger Datentypen)* mit denen der EM-Algorithmus einerseits eine Vielzahl von Modellen trainiert, andererseits aber das beste Modell als das *mit dem EM-Algorithmus trainierte Modell* festgelegt wird, sowie (iii) einen *Testkorpus (vollständiger Datentypen)* mit dem eine finale Evaluierung (für den Ergebnisreport) vorgenommen wird.

In Abschnitt 3.6 werden die grundlegenden Eigenschaften des EM-Algorithmus vorgestellt. Dabei wird schrittweise auf die Beantwortung der Frage hingearbeitet, ob die vom EM-Algorithmus generierten Modelle in wohldefinierter Weise gegen ein Modell streben, das eine *Maximum-Likelihood-Estimierung auf dem gegebenen Korpus der unvollständigen Datentypen* darstellt. Ausgangspunkt dieses Themas ist die Feststellung, dass die *Maximum-Likelihood-Estimierung unvollständiger Datentypen* ein mathematisch schwieriges Problem ist. Zur Vorbereitung für dessen Lösung wird einerseits gezeigt werden, dass die *Log-Likelihood der unvollständigen Datentypen* sich additiv aus der EM-Hilfsfunktion und der sogenannten *erwarteten Cross-Entropie aller Analysen* berechnet, andererseits werden einige wichtige *Theoreme aus der Differentialrechnung mehrerer Veränderlicher* vorgestellt, die bei der Maximierung von Funktionen mehrdimensionaler Parameter behilflich sind. Der originalen Arbeit von Dempster et al. (1977) folgend, wird der sogenannte *generalisierte EM-Algorithmus (GEM-Algorithmus)* vorgestellt und gezeigt, dass ein GEM-Algorithmus (unter sehr milden Voraussetzungen) eine Folge von Modellen mit monoton wachsender Log-Likelihood erzeugt (Theorem 1), deren Grenzwert zumindest ein *stationärer Punkt* (lokales Maximum, lokales Minimum, oder Sattelpunkt) der Log-Likelihood ist, sobald vorausgesetzt werden darf, dass der Grenzwert existiert und alle Modelle stationäre Punkte der EM-Hilfsfunktion sind (Theorem 4). Verbesserte Konvergenzaussagen zum EM-Algorithmus stammen von Wu (1983), der (ebenfalls unter Annahme milder Voraussetzungen) zeigt, dass an jedem *Häufigkeitspunkt* einer von einem GEM-Algorithmus ausgegebenen Parameterfolge ein *lokales Maximum der Log-Likelihood* vorliegt, wenn unter anderem vorausgesetzt werden kann, dass die Log-Likelihood für alle Glieder der Parameterfolge, die

keine lokalen Maxima sind, *streng monoton wächst* (Theorem 1). Unter nur geringfügig schärferen Voraussetzungen kann die diesen Abschnitt einleitende Frage positiv beantwortet werden und davon ausgegangen werden, dass *Parameter-Konvergenz gegen ein lokales Maximum der Log-Likelihood* vorliegt (Theorem 5). Der Abschnitt wird mit einigen eigenen Ergebnissen zum EM-Algorithmus geschlossen, die eine *injektive Parametriserung der Analysen* zum Thema haben und z.B. zeigen, dass der in einem GEM-Algorithmus re-estimierte Parameter einer Maximum-Likelihood-Estimierung *Parameter-invariant* ist, eine Eigenschaft, mit der sich sowohl Dempster et al. (1977), als auch Wu (1983) beschäftigt haben.

In Abschnitt 3.7 werden die ersten und zweiten partiellen Ableitungen der Log-Likelihood und der EM-Hilfsfunktion vorgestellt, um deren stationären Punkte berechnen und entscheiden zu können, ob es sich dabei um *lokale Maxima*, *lokale Minima*, oder *Sattelpunkte* handelt. Da beide Funktionen *logarithmische Terme* enthalten, wird der *Delta-Operator* eingeführt, mit dessen Hilfe alle nötigen analytischen Operationen gemäss den *Rechenregeln des Delta-Operators* stark vereinfacht werden können. Es wird sich überraschenderweise zeigen, dass sich die ersten partiellen Ableitungen der Log-Likelihood (und der EM-Hilfsfunktion) als Erwartungswerte der empirischen Wahrscheinlichkeitsverteilung für vollständige Datentypen darstellen lassen (siehe Abbildung 3.21). Ferner wird in diesem Abschnitt gezeigt, dass eine konvergente Folge von EM-Parametern ein lokales Maximum der Log-Likelihood als Grenzwert besitzt, wenn die *erwartete Cross-Entropie aller Analysen* in einer Umgebung dieses Punktes in einem bestimmten Sinn "kontrollierbar" ist.

In Abschnitt 3.8 wird der EM-Algorithmus auf probabilistische kontextfreie Grammatiken angewendet. Es wird sich zeigen, dass sich die *re-estimierten Grammatikregel-Wahrscheinlichkeiten* berechnen, indem die *erwartete Häufigkeit einer Grammatikregel* durch die *erwartete Häufigkeit ihrer Mutterkategorie* dividiert wird. Mit Hilfe dieser Re-Estimierungsformeln wird es in Abschnitt 3.9 möglich sein, den *Inside-Outside-Algorithmus* (Baker 1979) als eine *dynamische Programmiervariante des EM-Algorithmus* vorzustellen. Neben den Re-Estimierungsformeln werden auch sehr genau die Konvergenzeigenschaften dieser Instanz des EM-Algorithmus vorgestellt. Es wird sich zeigen, dass (i) die Log-Likelihood während des Trainings monoton wächst, und die Folge der re-estimierten Parameter gegen einen *stationären Punkt* der Log-Likelihood konvergiert. Falls vorausgesetzt werden kann, dass alle Mutterkategorien während der gesamten Re-Estimierung eine positive Vorkommenserwartung haben und die *erwartete Cross-Entropie aller Analysen* kontrollierbar ist, so liegt sogar Konvergenz gegen ein *lokales Maximum* der Log-Likelihood vor. Somit sind die Verhältnisse sehr viel komplexer, als in der Literatur gemeinhin angenommen wird. Zum Abschluss dieses Abschnitts wird gezeigt, dass das in Abschnitt 2.5

eingeführte *Baumbank-Trainingsverfahren* ein Spezialfall des EM-Algorithmus für annotierte Datentypen und probabilistische kontextfreie Grammatiken ist. Hierfür wird der *EM-Algorithmus für annotierte Datentypen* eingeführt und gezeigt, dass dieser Algorithmus den optimalen Parameter stets bereits nach einer einzigen Iteration findet. Angewandt auf eine Baumbank liefert der EM-Algorithmus die bekannten Re-Estimierungsformeln (siehe Abschnitt 2.5), sowie einen sehr kurzen Beweis der ausgezeichneten Eigenschaft, dass die so estimierten Grammatikregel-Wahrscheinlichkeiten die Baumbank-Wahrscheinlichkeit global maximieren.

In Abschnitt 3.9 werden die formalen Beweise der allseits bekannten Aussagen über den Inside-Outside-Algorithmus, die man entgegen der in der Linguistik weitverbreiteten Meinung (Manning und Schütze 1999) nicht in der Original-Literatur ((Baker 1979), (Lari und Young 1990)) und nach bestem Wissen des Autors auch nirgendwo sonst findet, nachgeholt werden. Hierbei handelt es sich um die Beweise für die folgenden Aussagen:

- Die Log-Likelihood des Inside-Outside-Algorithmus wächst während des Trainings mononton,
- Die Werte der Log-Likelihood des Inside-Outside-Algorithmus konvergieren gegen einen *stationären Wert*, in einigen Fällen sogar gegen ein *lokales Maximum*.

Für den Beweis dieser beiden Eigenschaften wird die folgende Aussage formal bewiesen:

- Der Inside-Outside-Algorithmus ist eine *dynamische Programmiervariante* des EM-Algorithmus für kontextfreie Grammatiken.

Auch diese Eigenschaft des Inside-Outside-Algorithmus ist in der Linguistik relativ gut bekannt, aber (nach bestem Wissen des Autors) ebenfalls noch nicht formal bewiesen worden. Zudem ist der exakte Beweis sehr viel weniger leicht, als üblicherweise angenommen wird (Manning und Schütze 1999) und erfordert tiefe Kenntnisse über den Inside-Outside-Algorithmus und den EM-Algorithmus. Der Schlüsselschritt des Beweises ist, dass die *Grammatikregel-Counts* aus dem Inside-Outside-Algorithmus formal (und nicht nur intuitiv) mit den *erwarteten Grammatikregel-Vorkommenshäufigkeiten* (auf der Menge der Syntaxbäume eines Satzes) im EM-Algorithmus identifiziert werden. Hierzu werden sowohl die Rekursionsformeln der Inside- und Outside-Wahrscheinlichkeiten benutzt werden, als auch die Rechenregeln der Erwartungswerte.

3.1 Der Inside-Outside-Algorithmus

Der Inside-Outside-Algorithmus wurde in seiner heute bekannten Form erstmals von Lari und Young (1990) vorgestellt, die nach einer Pause von ca. 10 Jahren, einen Algorithmus von Baker (1979) wiederaufgegriffen und anhand der dort vorgeschlagenen Re-Estimierungsformeln ein Trainingsverfahren für probabilistische kontextfreie Grammatiken vorstellten, welches nicht wie das bei Baker vorgeschlagene Verfahren auf einen einzigen Satz beschränkt ist, sondern prinzipiell auf *beliebig grossen (satzweise segmentierten) Textkorpora* arbeiten kann. Der von Lari und Young vorgestellte Algorithmus dient den meisten statistischen Parsern (abgesehen von internen, heuristisch benutzten Glättungsverfahren) als Implementierungsvorlage (siehe zum Beispiel Schmid (1999a)). In diesem Abschnitt soll der Inside-Outside-Algorithmus, in den wichtigsten Teilen der Darstellung von Lari und Young (1990) folgend, vorgestellt werden. Die Re-Estimierungsformeln des Inside-Outside-Algorithmus basieren auf den sogenannten Inside- bzw. Outside-Wahrscheinlichkeiten, die sich rekursiv (bottom-up bzw. top-down) berechnen lassen und dem Algorithmus seinen Namen gaben. In Abschnitt 3.9 wird formal gezeigt werden, dass der Inside-Outside-Algorithmus eine dynamische Programmiervariante des EM-Algorithmus ist und damit insbesondere die Konvergenzeigenschaften des EM-Algorithmus für kontextfreie Grammatiken erbt, welche ausführlich in Abschnitt 3.8 vorgestellt werden.

Die Besonderheit des Inside-Outside-Algorithmus ist, dass er ein iteratives Parameter-Estimierungsverfahren für probabilistische kontextfreie Grammatiken ist, welches keine annotierte Daten benötigt. Das andere bekannte Trainingsverfahren für probabilistische kontextfreie Grammatiken, das sogenannte Baumbank-Trainingsverfahren, wurde bereits in Abschnitt 2.5 vorgestellt. Es ist ein einfaches, nicht-iteratives Lernverfahren mit vielen ausgezeichneten Eigenschaften. Sein Nachteil ist jedoch, dass es eine Baumbank, d.h. einen grossen Korpus von manuell annotierten Sätzen voraussetzt. Die Annotationen sind in der Regel zudem relativ komplex, weil sie aus den von Hand disambiguierten Syntaxbäumen aller Sätze im Korpus bestehen. Mit dem Inside-Outside-Algorithmus werden die Grammatikregel-Wahrscheinlichkeiten für probabilistische kontextfreie Grammatiken ohne die Benutzung einer Baumbank re-estimiert. Ein Vorteil des Inside-Outside-Algorithmus ist daher, dass er mit freiem Text auskommt, der in der Regel für fast alle Sprachen, ohne Aufwand, in nahezu unbegrenztem Umfang zur Verfügung steht. Wie in Abschnitt 2.5 ausgeführt wurde, wird dieser Vorteil allerdings teilweise durch die Kosten einer manuellen Grammatik-Entwicklung kompensiert. Von einer Baumbank kann eine Grammatik direkt abgelesen werden (Charniak 1996). In der Arbeit von Ngai und Yarowsky (2000) werden weitere Vor- und Nachteile dieser zwei alternativen Vorgehensweisen diskutiert.

Aus der Sicht des Autors ist ein weiterer wesentlicher Vorteil des Inside-Outside-Algorithmus, dass er im Vergleich zum Baumbank-Trainingsverfahren sehr viel mehr sogenannte *freie Parameter* besitzt. Der einzige freie Parameter des Baumbank-Trainingsverfahrens ist die Grösse der verwendeten Baumbank. Weil der Inside-Outside-Algorithmus ein iteratives Trainingsverfahren ist, welches mit einer zufällig initialisierten Startgrammtik beginnt, d.h. mit zufällig initialisierten Grammatikregel-Wahrscheinlichkeiten, hat er wesentlich mehr freie Parameter als das Baumbank-Trainingsverfahren:

- die *Grösse des Trainingskorpus*,
- die *Anzahl der Trainingsiterationen*,
- die *Startgrammatik*.

Die Schwierigkeiten im Umgang mit diesen Parametern werden in der Literatur oft als Nachteil des Inside-Outside-Algorithmus angesehen. Das Gegenteil ist der Fall: Eine systematische Variation der freien Parameter hat zur Folge, dass der Inside-Outside-Algorithmus im Gegensatz zum Baumbank-Trainingsverfahren (bei festgehaltener Trainingskorpusgrösse) nicht nur eine einzige, sondern viele, stochastisch wohlfundierte, probabilistische kontextfreie Grammatiken ausgibt. Unter allen vom Inside-Outside-Algorithmus "vorgeschlagenen stochastifizierten Grammatiken" kann im allgemeinen stets eine Grammatik ausgewählt werden, die in der linguistischen Evaluierung gute Ergebnisse zeigt. Da die Erfahrung zeigt, dass die linguistischen Evaluierungsmetriken in der Regel nicht mit den Likelihood-Werten korrelieren (Beil et al. 1998), ist dies ein sehr wichtiges Argument für den Inside-Outside-Algorithmus, bzw. gegen *Baumbank-Grammtiken*, die ja ausschliesslich mit Blick auf die Baumbank-Likelihood hergestellt werden (siehe Abschnitt 2.5).

Es sollte überflüssig sein, ausdrücklich festzuhalten, dass der Inside-Outside-Algorithmus nur bei einer einzigen Vorgehensweise gute Ergebnisse zeigen kann: *Alle freien Parameter des Inside-Outside-Algorithmus müssen auf der Suche nach der linguistisch besten Grammatik systematisch variiert werden.* In der Literatur liest man immer wieder, dass mit dem Inside-Outside-Algorithmus in der Regel nur schlechte Ergebnisse zu erzielen sind. Leider stellt man genauso oft fest, dass in denselben Arbeiten die freien Parameter des Inside-Outside-Algorithmus garnicht, oder nur unvollständig variiert und evaluiert wurden: oft betrifft dies die Variation der Trainingskorpusgrösse, öfter die Iterationsanzahl, am häufigsten die Startwerte. Dies ist unverständlich, weil es viele neuere Arbeiten gibt, die zeigen, dass sowohl eine Variation der Startwerte, als auch der Trainingsiterationen jeweils bis zu 30% Gewinn in der *linguistischen Evaluierung* bringen kann (Müller 2000). Beide Parameter haben in der Regel einen sehr viel grösseren Einfluss, als die in ihrer Wichtigkeit vielleicht überschätzte Variation der Trainingskorpusgrösse. Obwohl es in

jüngster Zeit in alternativen Ansätzen gelang, theoretisch wohl-fundierte Startwerte anzugeben, die zudem sehr gute praktische Resultate zeigen (Riezler et al. 2000), sind dem Autor keine theoretischen Aussagen zu guten Startwerten für probabilistische kontextfreie Grammatiken bekannt. In der Praxis ist es leider nicht unüblich, die Gleichverteilung (für Grammatikregeln mit derselben linken Seite) als feste Startgrammatik zu benutzen. Dieses Vorgehen ist wahrscheinlich dadurch motiviert, sehr viel Rechenzeit einzusparen. Dieses theoretisch unmotivierte Verfahren hat in der Regel schlechte Ergebnisse zur Folge, da ein Freiheitsgrad des Inside-Outside-Algorithmus vollkommen unausgenutzt bleibt.

Dieser Abschnitt ist wie folgt gegliedert. Zunächst werden die Inside-Parsewälder und -Wahrscheinlichkeiten einer erneuten Betrachtung unterzogen, dann werden die Outside-Parsewälder und -Wahrscheinlichkeiten definiert. Anschliessend werden die Re-Estimierungsformeln des Inside-Outside-Algorithmus vorgestellt, um den Abschnitt mit Betrachtungen zur Evaluierung der mit dem Inside-Outside-Algorithmus trainierten Grammatiken abzuschliessen.

Inside-Parsewälder- und Wahrscheinlichkeiten

Die *Outside-Parsewälder* und *-Wahrscheinlichkeiten* sind in gewissem Sinn die Gegenstücke der in Abschnitt 2.6 eingeführten *(Inside-)Parsewälder* und *Inside-Wahrscheinlichkeiten*. Es ist deshalb vorteilhaft, sich zunächst deren Definitionen in Erinnerung zu rufen. In Abschnitt 2.6 wurde der sogenannte Parsewald-Algorithmus (Abbildung 2.11) vorgestellt, mit dem für eine probabilistische kontextfreie Grammatik G in Chomsky-Normalform die Parsewälder:

$$\begin{aligned} forest[s, A, s+l] &:= A \Rightarrow^* w_s \dots w_{s+l-1} \\ &:= \left\{ x \in \mathcal{T}(G) \;\middle|\; \begin{array}{l} \text{root}(x) = A, \\ \text{yield}(x) = w_s \dots w_{s+l-1} \end{array} \right\} . \end{aligned}$$

dynamisch[1] und daher effizient berechnet werden können. Hierbei ist A eine beliebige Grammatikkategorie, $w = w_1 \dots w_n$ ein vorgegebener Satz und $w_s \dots w_{s+l-1}$ ein beliebiger Teilsatz der Länge l. In der Definition bezeichnet

$$\text{root}(x)$$

die Grammatikkategorie, und

$$\text{yield}(x)$$

die Blattknotenfolge eines Syntaxbaumes x aus der Menge $\mathcal{T}(G)$ aller möglichen Syntaxbäume der Grammatik G. Mit dem Inside-Algorithmus (Abbildung 2.14) werden die

[1] "Dynamische Programmiertechniken" werden z.B. in Cormen et al. (1994) vorgestellt.

Inside-Wahrscheinlichkeiten

$$inside[s, A, s+l] \;:=\; p(A \Rightarrow^* w_s \ldots w_{s+l-1})$$

dynamisch und effizient berechnet. Gemäss dieser Definition ist die Inside-Wahrscheinlichkeit eines Teilsatzes die Summe der Wahrscheinlichkeiten aller Syntaxbäume dieses Teilsatzes, die die Wurzelkategorie A tragen. In Abschnitt 2.7 wurde der Semiring-Parsing-Algorithmus vorgestellt und gezeigt, dass der Parsewald- und der Inside-Algorithmus, wie auch viele weitere stochastische Parsingverfahren, einfache Ausprägungen dieses dynamischen Algorithmus sind. Hierbei ist äusserst interessant, dass die Möglichkeit zur effizienten *dynamischen Programmierung* des Semiring-Parsing-Algorithmus lediglich auf zwei Faktoren beruht: (i) die rekursive Berechnung der Parsewälder ist möglich, (ii) die Berechnung interessanter stochastischer Werte ergibt sich mittels *Homomorphie* aus der Berechnung der Parsewälder.

Für die Darstellung des Semiring-Parsing-Algorithmus waren die Bezeichnungen aus Goodman (1998) sehr zweckmässig. Um aber der Darstellung des Inside-Outside-Algorithmus von Lari und Young (1990) besser folgen zu können, werden die Inside-Parsewälder

$$\mathcal{F}_{\text{inner}}(s, t, A) \;=\; A \Rightarrow^* w_s \ldots w_t$$

und die Inside-Wahrscheinlichkeiten

$$e(s, t, A) \;=\; p(A \Rightarrow^* w_s \ldots w_t) \;=\; p\left(\mathcal{F}_{\text{inner}}(s, t, A)\right)$$

neu bezeichnet ($A \in G,\ 1 \leq s \leq t \leq n$). Für einen Teilstring der Länge $l = 1 + t - s$ gilt dann der folgende Zusammenhang zwischen den alten und neuen Bezeichnungen:

$$\mathcal{F}_{\text{inner}}(s, t, A) = forest[s, A, s+l], \qquad e(s, t, A) = inside[s, A, s+l].$$

Abbildung 3.1 motiviert (unter Benutzung der neu eingeführten Bezeichnungen), dass sich die Inside-Parsewälder rekursiv berechnen könnten. Es ist nicht schwer, für die Inside-Parsewälder und -Wahrscheinlichkeiten die folgenden Rekursionsformeln zu beweisen (siehe auch Abschnitt 2.6).

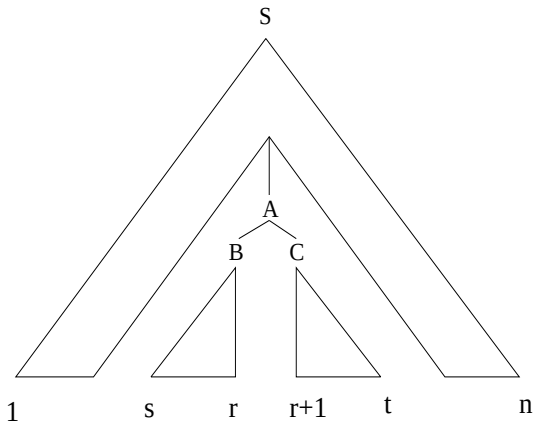


Abbildung 3.1: Rekursion für Inside-Parsewälder und Wahrscheinlichkeiten

Rekursion für Inside-Parsewälder und Wahrscheinlichkeiten:

$(1 \leq s \leq n)$:

$$\mathcal{F}_{\text{inner}}(s,s,A) \;=\; \begin{cases} \{\; A \to w_s \;\} & \text{falls } (A \to w_s) \in G \\ \emptyset & \text{sonst} \end{cases}$$

und

$$e(s,s,A) \;=\; \begin{cases} p(A \to w_s) & \text{falls } (A \to w_s) \in G \\ 0 & \text{sonst} \end{cases}$$

$(1 \leq s < t \leq n)$:

$$\mathcal{F}_{\text{inner}}(s,t,A) \;=\; \sum_{A \to BC} \sum_{r=s}^{t-1} \{A \to B\ C\} \;\otimes\; \mathcal{F}_{\text{inner}}(s,r,B) \;\otimes\; \mathcal{F}_{\text{inner}}(r+1,t,C)$$

und

$$e(s,t,A) \;=\; \sum_{A \to BC} \sum_{r=s}^{t-1} p(A \to B\ C) \;\cdot\; e(s,r,B) \;\cdot\; e(r+1,t,C)\ .$$

In den Formeln steht das Symbol '$\otimes$' für die Multiplikation von Parsewäldern innerhalb des Parsewald-Semirings (siehe Abbildung 2.22), während das Summationssymbol $\sum$ auf die Vereinigung disjunkter Mengen verweist, die im Parsewald-Semiring als Addition dient. Details entnimmt man Abschnitt 2.7.

Die Inside-Parsewälder und -Wahrscheinlichkeiten können daher rekursiv berechnet werden, indem zunächst die Werte $\mathcal{F}_{\text{inner}}(.)$ bzw. $e(.)$ für alle Teilsätze der Länge 1, dann für die Teilsätze der Länge 2, usw. bestimmt werden (*Bottom-Up-Berechnung*).

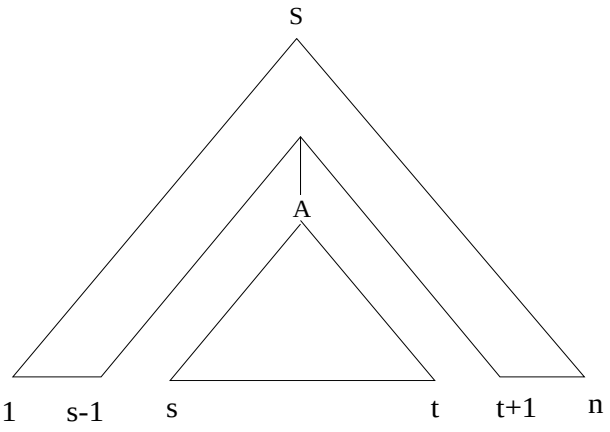


Abbildung 3.2: Definition des Outside-Parsewald

Outside-Parsewälder und -Wahrscheinlichkeiten

Im folgenden werden die *Outside-Parsewälder* und *-Wahrscheinlichkeiten* definiert. Per Definition gilt (für $1 \leq s \leq t \leq n$):

$$\begin{aligned} \mathcal{F}_{\text{outer}}(s,t,A) &:= S \Rightarrow^* w_1 \dots w_{s-1}\ A\ w_{t+1} \dots w_n \\ &= \left\{ x \in \mathcal{T}(G) \;\middle|\; \begin{array}{l} \text{root}(x) = S, \\ \text{yield}(x) = w_1 \dots w_{s-1}\ A\ w_{t+1} \dots w_n \end{array} \right\} . \end{aligned}$$

Abbildung 3.2 zeigt den Outside-Parsewald $\mathcal{F}_{\text{outer}}(s,t,A)$ zusammen mit dem Inside-Parsewald $\mathcal{F}_{\text{inner}}(s,t,A)$. Es ist bemerkenswert, dass ein Outside-Parsewald im Gegensatz zu einem Inside-Parsewald $\mathcal{F}_{\text{inner}}(s,t,A)$ nicht durch die Angabe eines Teilsatzes $w_s \dots w_t$ und einer Grammatikkategorie definiert ist. Vielmehr muss zur Definition eines Outside-Parsewalds neben der Grammatikkategorie A deren linker und rechter Satzkontext, $w_1 \dots w_{s-1}$ und $w_{t+1} \dots w_n$, angegeben werden. Die *Outside-Wahrscheinlichkeiten* sind, ebenso wie die Inside-Wahrscheinlichkeiten, als die Wahrscheinlichkeiten der entsprechenden Parsewälder definiert:

$$f(s,t,A) := p\left(\mathcal{F}_{\text{outer}}(s,t,A)\right) .$$

Die Abbildung 3.2 macht bereits deutlich, dass gewisse Inside- und Outside-Parsewälder den Parsewald des vollständigen Satzes generieren werden. Bevor diese Beobachtung detaillierter behandelt wird, soll gezeigt werden, dass auch die Outside-Parsewälder und -Wahrscheinlichkeiten rekursiv berechnet werden können.

Man kann sich vorstellen, dass der Parsewald $\mathcal{F}_{\text{outer}}(s,t,A)$ durch einen Ableitungsprozess entstanden ist, in welchem die Grammatikkategorie A erzeugt wurde, und es zwei Teilsätze

$$w_1 \dots w_{s-1}$$

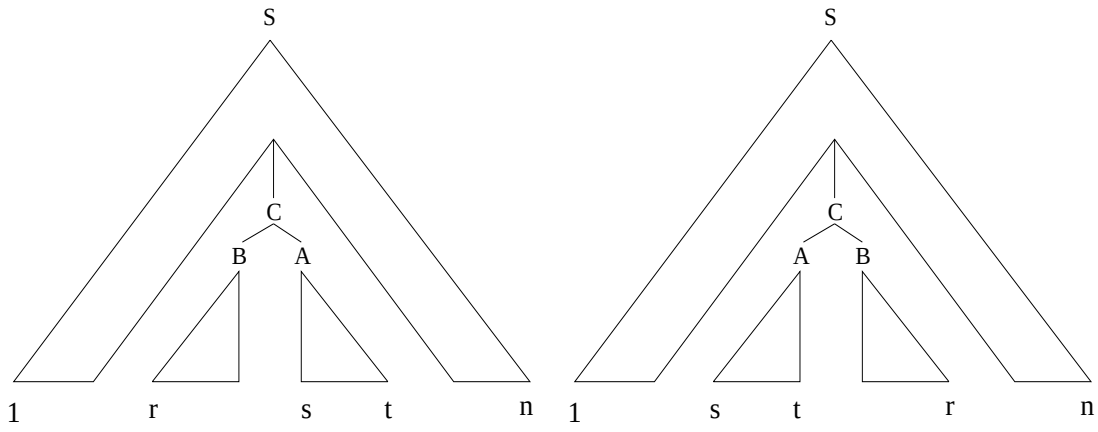


Abbildung 3.3: Rekursion für Outside-Parsewälder und Wahrscheinlichkeiten

(zur linken Seite der Grammatikkategorie) und

$$w_{t+1} \dots w_n$$

(zur rechten Seite) gibt, die nicht von A dominiert werden. In diesem Fall wird es eine Grammatikkategorie C geben, sodass eine der beiden Fälle $C \to BA$ oder $C \to AB$ eintritt (siehe Abbildung 3.3).

Rekursion für Outside-Parsewälder und Wahrscheinlichkeiten:

$(s = 1,\ t = n)$:

$$\mathcal{F}_{\text{outer}}(s,t,A) \;=\; \begin{cases} 1_{Parsewald} & \text{falls } A = S \\ 0_{Parsewald} & \text{sonst} \end{cases}$$

und

$$f(s,t,A) \;=\; \begin{cases} 1 & \text{falls } A = S \\ 0 & \text{sonst} \end{cases}$$

$(1 \le s \le t \le n$ mit $1 < s$ oder $t < n)$:

$$\mathcal{F}_{\text{outer}}(s,t,A) \;=\; \sum_{B,\ C \in G} \Bigg(\sum_{r=1}^{s-1} \mathcal{F}_{\text{outer}}(r,t,C) \;\otimes\; \begin{array}{ccc} & C & \\ \mathcal{F}_{\text{inner}}(r,s-1,B) & & A \end{array}$$

$$+ \sum_{r=t+1}^{n} \mathcal{F}_{\text{outer}}(s,r,C) \;\otimes\; \begin{array}{ccc} & C & \\ A & & \mathcal{F}_{\text{inner}}(t+1,r,B) \end{array} \Bigg)$$

und

$$f(s,t,A) \;=\; \sum_{B,\ C \in G} \Bigg(\sum_{r=1}^{s-1} f(r,t,C) \cdot p(C \to BA) \cdot e(r,s-1,B) + \sum_{r=t+1}^{n} f(s,r,C) \cdot p(C \to AB) \cdot e(t+1,r,B) \Bigg)$$

Natürlich müssen hierbei die Parsewälder

$$\begin{array}{ccc} & C & \\ \mathcal{F}_{\text{inner}}(r,s-1,B) & & A \end{array} \quad \text{und} \quad \begin{array}{ccc} & C & \\ A & & \mathcal{F}_{\text{inner}}(t+1,r,B) \end{array}$$

mit der leeren Menge identifiziert werden, wenn es keine entsprechenden Grammatikregeln

$$C \to BA \text{ oder } C \to AB$$

gibt. Nach der *Bottom-Up-Berechnung* der Inside-Parsewälder und -Wahrscheinlichkeiten kann daher eine *Top-Down-Berechnung* der Outside-Parsewälder und -Wahrscheinlichkeiten erfolgen.

Inside- und Outside-Wahrscheinlichkeiten im Inside-Outside-Algorithmus

Das Produkt

$$\mathcal{F}_{\text{outer}}(s,t,A) \otimes \mathcal{F}_{\text{inner}}(s,t,A)$$

ist im Parsewald-Semiring wohldefiniert. Der Abbildung 3.2 entnimmt man, dass der Inside-Parsewald vermittels der Grammatikkategorie A in den Outside-Parsewald eingehängt wird. Für je zwei fixierte Indizes $s \leq t$ gilt dann:

$$\sum_{A \in G} \mathcal{F}_{\text{outer}}(s,t,A) \otimes \mathcal{F}_{\text{inner}}(s,t,A) = A \Rightarrow^* w_1 \dots w_n \ ,$$

bzw.

$$\sum_{A \in G} f(s,t,A) \cdot e(s,t,A) = p(A \Rightarrow^* w_1 \dots w_n) \ .$$

Diesen Formeln kann entnommen werden, dass es möglich ist, den Parsewald und die Wahrscheinlichkeit des Satzes $w_1 \dots w_n$ nur unter Benutzung der Inside-Werte (setze $s = 1,\ t = n$) oder der Outside-Werte (setze $s = t$) zu berechnen, was für Parsingzwecke nützlich ist (siehe auch Abschnitt 2.6).

Der Outside-Algorithmus

Abbildung 3.4 zeigt den *Outside-Algorithmus*. Er ist wie sein Pendant, der Inside-Algorithmus (Abschnitt 2.6), ein stochastischer Parsing-Algorithmus. Seine Eingabe besteht aus einer probabilistischen kontextfreien Grammatik in Chomsky-Normalform, einem Satz $w = w_1 \dots w_n,\ (n \geq 1)$, sowie den Inside-Wahrscheinlichkeiten $inside[s, A, s+l]$ dieses Satzes. Die Ausgabe des Outside-Algorithmus besteht aus den Outside-Wahrscheinlichkeiten

$$outside[s, A, s+l] \ = \ p(S \Rightarrow^* w_1 \dots w_{s-1} A w_{s+l} \dots w_n \Rightarrow^* w_1 \dots w_n)$$

für die Teilsätze der *Länge* $1 \leq l \leq n$ und deren *Startindizes* $1 \leq s \leq n+1-l$. Offensichtlich ist diese Definition an die Definition der Inside-Wahrscheinlichkeiten aus Abschnitt 2.6 angelehnt. Der Outside-Algorithmus beruht wie der Semiring-Parsing-Algorithmus auf dynamischer Programmierung und benutzt die oben angebenen Rekursionsformeln für Outside-Wahrscheinlichkeiten. In der 1. Zeile der Prozedur des Outside-Algorithmus werden die Outside-Wahrscheinlichkeiten mit 0 initialisiert. In der 2. Zeile wird die Wahrscheinlichkeit des leeren (Outside-) Parsewalds mit dem neutralen Element der Multiplikation initialisiert. Die eigentliche Rekursion findet in der 3. bis 10. Zeile der Prozedur statt. Offensichtlich ist der Outside-Algorithmus wie der Inside-Algorithmus von der Ordnung $O(n^3)$.

Eingabe: Eine probabilistische kontextfreie Grammatik in Chomsky-Normalform, ein Satz $w = w_1 \dots w_n$ und dessen Inside-Wahrscheinlichkeiten $inside[...]$ (berechnet mit dem Inside-Algorithmus aus Abbildung 2.14).

Ausgabe: Die Outside-Wahrscheinlichkeiten $outside[s, A, s + l]$ für alle Teilsätze der Länge $1 \leq l \leq n$, Startindizes $1 \leq s \leq n + 1 - l$, und Grammatikkategorien A.

Prozedur:

1. $outside[., ., .] := 0$;
2. $outside[1, S, n + 1] := 1$;
3. **for each** $l := n, \dots, 2$ **do**
4. **for each** $s := 1, \dots, n + 1 - l$ **do**
5. **for each** $t := 1, \dots, l - 1$ **do**
6. **for each** $A \to B\ C$ **do**
7. $outside[s, B, s + t] := outside[s, B, s + t] +$
8. $p(A \to B\ C) \cdot outside[s, A, s + l] \cdot inside[s + t, C, s + l]$;
9. $outside[s + t, C, s + l] := outside[s + t, C, s + l] +$
10. $p(A \to B\ C) \cdot outside[s, A, s + l] \cdot inside[s, B, s + t]$;

Abbildung 3.4: Outside-Algorithmus

Der Inside-Outside-Algorithmus in der Version von Baker (1979)

Es ist allgemein bekannt, dass in der Arbeit von Baker (1979) erstmals Re-Estimierungformeln für die Grammatikregel-Wahrscheinlichkeiten einer probabilistischen kontextfreien Grammatik (in Chomsky-Normalform) vorgestellt wurden. Es ist aber wenig bekannt, dass Baker die Re-Estimierungsformeln lediglich für einen Spezialfall angab, der leider für die praktische Anwendung bedeutungslos ist, nämlich für den Fall, dass der Trainingskorpus nur aus einem einzigen Satz besteht.

Es kann vermutet werden, dass Baker allerdings schon mit diesem Spezialfall (aus heutiger Sicht) zufrieden war. Sein Ziel war lediglich, die Re-Estimierungsformeln für die Startzustands- und Zustandsübergangs-Wahrscheinlichkeiten eines Hidden-Markov-Modells zu verallgemeinern, und die Hidden-Markov-Modelle durch die in der *Chomsky-Hierarchie* nächstmächtigeren kontextfreien Grammatiken zu ersetzen. Aus der Arbeit von 1979 ist ersichtlich, dass Baker die neuen Re-Estimierungsformeln als Ergebnis eines rein *intuitiven Abstraktionsprozesses* erhielt, indem er die Formeln von Baum (1972) als Ausgangspunkt nahm. Im Bewusstsein, dass jedes Hidden-Markov-Modell auf einem Trainingskorpus von Worten trainiert wird, ging Baker vielleicht davon aus, dass auch der Trainingskorpus einer probabilistischen kontextfreien Grammatik stets aus einer Wortsequenz, also aus einem einzigen Satz bestehen sollte. Unter dieser vereinfachenden Annahme erhielt er die folgenden Re-Estimierungsformeln:

Baker (1979): Re-Estimierungsformeln für kontextfreie Grammatiken und Trainingskorpora mit einem einzigen Satz: Ist $w_1 \dots w_n$ der einzige Satz eines gegebenen Trainingskorpus, so kann aus einer kontextfreien Grammatik (in Chomsky-Normalform) eine probailistische kontextfreie Grammatik trainiert werden, indem die Grammatikregel-Wahrscheinlichkeiten (ausgehend von zufälligen Grammatikregel-Wahrscheinlichkeiten) in iterativen *Re-Estimierungsschritten* berechnet werden. In jedem Re-Estimierungsschritt berechnen sich die *re-estimierten Grammatikregel-Wahrscheinlichkeiten* $\hat{p}(r),\ r \in G$ aus den gegenwärtigen Grammatikregel-Wahrscheinlichkeiten $p(r),\ r \in G$, wie folgt:

$$\hat{p}(A \to a) \ = \ \frac{\frac{1}{P} \sum_{1 \le t \le n,\ w_t = a} e(t,t,A) \cdot f(t,t,A)}{\frac{1}{P} \sum_{s=1}^{n} \sum_{t=s}^{n} e(s,t,A) \cdot f(s,t,A)} \ ,$$

und

$$\hat{p}(A \to BC) = \frac{\frac{1}{P} \sum_{s=1}^{n-1} \sum_{t=s+1}^{n} \sum_{r=s}^{t-1} p(A \to BC) e(s,r,B) e(r+1,t,C) f(s,t,A)}{\frac{1}{P} \sum_{s=1}^{n} \sum_{t=s}^{n} e(s,t,A) f(s,t,A)} \ .$$

Hierbei wird die Satzwahrscheinlichkeit

$$P = p(S \Rightarrow^* w_1 \dots w_n)$$

und die Inside-Wahrscheinlichkeiten (mit dem Inside-Algorithmus), sowie die Outside-Wahrscheinlichkeiten (mit dem Outside-Algorithmus) aus den gegenwärtigen Grammatikregel-Wahrscheinlichkeiten berechnet.

Es ist interessant, dass sich in der Arbeit von Baker im Gegensatz zu Baum (1972) keine Aussage (und daher auch kein Beweis) zur Monotonie der Log-Likelihood während der iterierten Anwendung der Re-Estimierungsschritte findet.

Bakers Re-Estimierungsformeln für probabilistische kontexfreie Grammatiken sind rein intuitiv erhaltene Verallgemeinerungen von Baums Re-Estimierungsformeln für Hidden-Markov-Modelle.

Möglicherweise nahm Baker deshalb an, dass die von ihm vorgenommenen Verallgemeinerungen der Re-Estimierungsformeln bewirken würden, dass sich auch die Eigenschaften der re-estimierten Hidden-Markov-Modelle (Monotonie der Log-Likelihood) auf die re-estimierten probabilistischen kontextfreien Grammatiken übertragen, ohne es formal beweisen zu müssen.

Nach Ansicht des Autors kann diese Annahme allerdings kein Ersatz für einen formalen Beweis sein, dass die Log-Likelihood des Trainingskorpus kontextfreier Grammatiken mo-

noton wächst, zumal in der modernen Version dieser Aussage zusätzlich davon ausgegangen wird, dass ein Trainingskorpus nicht nur einen, sondern viele tausend Sätze enthält.

Ferner ist der in Abschnitt 3.9 vorgestellte Beweis relativ komplex, und setzt sowohl gute Kennnisse des EM-Algorithmus, als auch der Dynamik kontexfreier Grammatiken voraus. Es erscheint daher unangemessen, zu behaupten, dieser Beweis sei "intuitiv klar" und dürfe vorausgesetzt werden.

Der Inside-Outside-Algorithmus in der Version von Lari und Young (1990)

Damit die Re-Estimierungsformeln von Baker (1979) in praktischen Anwendungen benutzt werden konnten, war es nötig, die Re-Estimierungsformeln auf Trainingskorpora beliebiger Grösse zu verallgemeinern. Dies ist der Verdienst von Lari und Young (1990), die den Inside-Outside-Algorithmus ca. 10 Jahre nach der grundlegenden Arbeit von Baker zum ersten Mal in der heute modernen Form präsentierten.

Der Schlüsselschritt ist die Benutzung sogenannter *Grammatikregel- und Kategorie-Counts*. Für einen Satz $w = w_1 \dots w_n$ präsentieren sich die Re-Estimierungsformeln von Baker (1979) unter Benutzung von

$$C_w(A \to a) \;:=\; \frac{1}{P} \sum_{1 \le t \le n,\ w_t = a} e(t,t,A) \cdot f(t,t,A)$$

und

$$C_w(A) \;:=\; \frac{1}{P} \sum_{s=1}^{n} \sum_{t=s}^{n} e(s,t,A) \cdot f(s,t,A)\ ,$$

sowie

$$C_w(A \to BC) \;:=\; \frac{1}{P} \sum_{s=1}^{n-1} \sum_{t=s+1}^{n} \sum_{r=s}^{t-1} p(A \to BC) e(s,r,B) e(r+1,t,C) f(s,t,A)$$

in der folgenden einfachen Form:

$$\hat{p}(A \to a) \;=\; \frac{C_w(A \to a)}{C_w(A)}, \quad \hat{p}(A \to BC) \;=\; \frac{C_w(A \to BC)}{C_w(A)}.$$

Die Idee von Lari und Young (1990) ist sehr einfach. Die *Grammtikregel- und Kategorie-Counts* werden für einen gegebenen Trainingskorpus von Sätzen satzweise berechnet, sowie über dem gesamten Trainingskorpus akkumuliert, um daraus re-estimierte Grammatikregel-Wahrscheinlichkeiten zu berechnen.

Lari und Young (1990): Re-Estimierungsformeln für kontextfreie Grammatiken: Sind $y_1 \dots y_N$ die Sätze eines gegebenen Trainingskorpus, so kann aus einer kontextfreien Grammatik (in Chomsky-Normalform) eine probailistische kontextfreie Grammatik trainiert werden, indem die Grammatikregel-Wahrscheinlichkeiten (ausgehend von zufälligen Grammatikregel-Wahrscheinlichkeiten) in iterativen *Re-Estimierungsschritten* berechnet werden. In jedem dieser Re-Estimierungsschritte berechnen sich die *re-estimierten Grammatikregel-Wahrscheinlichkeiten* $\hat{p}(r),\ r \in G$ aus den gegenwärtigen Grammatikregel-Wahrscheinlichkeiten $p(r),\ r \in G$:

$$\hat{p}(A \to a) \ = \ \frac{\sum_{w=y_1}^{y_N} C_w(A \to a)}{\sum_{w=y_1}^{y_N} C_w(A)} \ ,$$

und

$$\hat{p}(A \to BC) = \frac{\sum_{w=y_1}^{y_N} C_w(A \to BC)}{\sum_{w=y_1}^{y_N} C_w(A)} \ .$$

Hierbei werden die *Grammatikregel- und Kategorie-Counts*

$$C_w(A \to a),\ C_w(A),\ C_w(A \to BC),$$

satzweise mit dem Inside- und dem Outside-Algorithmus aus den gegenwärtigen Grammatikregel-Wahrscheinlichkeiten berechnet.

Es ist erstaunlich, dass auch Lari und Young (1990) keinerlei Konvergenzbeweise für diese neuen, auf beliebige Trainingskorpora anwendbare Re-Estimierungsformeln vorstellen.

Es ist ferner interessant, dass sie wie selbstverständlich davon ausgehen, dass die neuen Re-Estimierungsformeln die Log-Likelihood eines Trainingskorpus schrittweise erhöhen werden, aber (wie beim Training von Hidden-Markov-Modellen) im allgemeinen keine Konvergenz gegen ein *globales Maximum* vorliegen wird.

Ferner ist erwähnenswert, dass Lari und Young in Experimenten die Log-Likelihood explizit für bis zu 100 Iterationen berechnen und grafisch zeigen, dass die Log-Likelihood monoton steigt (Lari und Young (1990), Abbildung 5, Seite 47).

Der Inside-Outside-Algorithmus in prozeduraler Form

Abbildung 3.5 zeigt den Inside-Outside-Algorithmus in prozeduraler Form. Die dort gezeigte Darstellung folgt weitgehend der Darstellung des Inside-Outside-Algorithmus von Lari und Young (1990). Es gibt drei Unterschiede: (i) Lari und Young (1990) behaupteten nicht explizit (und bewiesen es auch nicht), dass die probabilistischen kontextfreien Grammatiken mononton wachsende Log-Likelihood-Werte des Trainingskorpus produzieren. Kon-

Eingabe: Eine *Startgrammatik*, d.h. eine beliebige probabilistische kontextfreie Grammatik G_0 in Chomsky-Normalform, ein Trainingskorpus von Sätzen: $\{y_i \mid 1 \leq i \leq numberOfSentences\}$, sowie eine Iterationsanzahl $numberOfIterations \geq 1$.

Ausgabe: Eine Folge von *trainierten* probabilistischen kontextfreien Grammatiken G_i, $(1 \leq i \leq numberOfIterations)$; Die zugehörige Folge der Log-Likelihood-Werte des Trainingskorpus wächst monoton und konvergiert gegen einen *stationären Wert der Log-Likelihood-Funktion* (siehe Abschnitt 3.9).

Prozedur: *** Mit externer Evaluierung ***\

1. **for each** $i := 1, \ldots, numberOfIterations$ **do**
2. $C[.] := 0$;
3. $p(.) := p_{G_{i-1}}(.)$;
4. **for each** $j := 1, \ldots, numberOfSentences$ **do**
5. $w := y_j$;
6. **compute** $inside[.,.,.]$; $outside[.,.,.]$;
7. **for each** $A \rightarrow \alpha$ **do**
8. **compute** $C_w(A \rightarrow \alpha)$; $C_w(A)$;
9. **compute** total rule and category counts
   $C[A \rightarrow \alpha] = C[A \rightarrow \alpha] + C_w(A \rightarrow \alpha)$; $C[A] = C[A] + C_w(A)$;
10. **for each** $A \rightarrow \alpha$ **do**
11. $p_{G_i}(A \rightarrow \alpha) := \frac{C[A \rightarrow \alpha]}{C[A]}$;
12. **print** G_i;

Abbildung 3.5: Inside-Outside-Algorithmus

vergenz gegen stationäre Werte war damals ebenfalls unbekannt. Ein Beweis dieser beiden Aussagen findet sich in Abschnitt 3.9. (ii) Lari und Young (1990) benutzten keine feste Anzahl von Iterationen (*numberOfIterations*), sondern sahen vor, dass die Re-Estimierung abgebrochen wird, sobald sich in einem Re-Estimierungsschritt die Grammatikregel-Wahrscheinlichkeiten der gegenwärtigen Grammatik und der re-estimierten Grammatik höchstens um einen vorgegebenen Schwellenwert unterscheiden. (iii) Lari und Young (1990) gaben nicht die Folge aller trainierten Grammatiken aus, sondern sie arbeiteten nur mit der letzten Grammatik.

Leider haben viele neuere Experimente (Beil et al. (1998), Beil et al. (1999), Müller (2000)) gezeigt, dass dieses Vorgehen prinzipiell mangelhaft ist. In diesen Arbeiten wurde experimentell gezeigt, dass die Ergebnisse einer Evaluierung über die Log-Likelihood eines (Heldout-) Korpus in der Regel nicht mit den Ergebnissen einer linguistischen Evaluierung korrelieren. Somit wird eine gemäss Lari und Young (1990) "austrainierte Grammatik" (bei der ein weiterer Re-Estimierungsschritt keine grösseren Veränderungen der Grammatikregel-Wahrscheinlichkeiten nach sich zieht) *erfahrungsgemäss sehr schlechte Ergebnisse in der linguistischen Evaluierung und Anwendung aufweisen.* Daher ist die in Abbildung 3.5 gezeigte Variante des Inside-Outside-Algorithmus gegenüber der Variante von Lari und Young (1990) zu bevorzugen.

Die Eingabe des Inside-Outside-Algorithmus besteht aus einer sogenannten *Startgrammatik*, d.h. einer beliebigen probabilistischen Grammatik (in Chomsky-Normalform), einem Trainingskorpus $\{y_i \mid 1 \leq i \leq numberOfSentences\}$, sowie einer Iterationsanzahl $numberOfIterations \geq 1$. Die Ausgabe des Inside-Outside-Algorithmus besteht aus den sogenannten *trainierten* probabilistischen Grammatiken G_i, $(1 \leq i \leq numberOfIterations)$. In Abschnitt 3.9 wird gezeigt werden, dass die zugehörige Folge der Log-Likelihood-Werte des Trainingskorpus monoton wächst und gegen einen stationären Wert der Log-Likelihood konvergiert. In der 1. Zeile der Prozedur des Inside-Outside-Algorithmus wird eine Schleife über die Trainingsiterationen initialisiert, deren Schritte in der 2. bis 12. Zeile ausgeführt werden. In der 2. Zeile werden die *totalen Grammatikregel- und Kategorie-Counts* $C[.]$ mit Null initialisiert. In der 3. Zeile wird festgelegt, dass in den folgenden Schritten die Grammatikregel-Wahrscheinlichkeiten der in der letzten Trainingsiteration hergestellten Grammatik benutzt werden. In der 4. Zeile wird eine Schleife über die Sätze des Trainingskorpus initialisiert, deren Schritte in der 5. bis 9. Zeile ausgeführt werden. Hierbei werden die *Grammatikregel- und Kategorie-Counts* des jeweils aktuellen Satzes berechnet und zu den *totalen Grammatikregel- und Kategorie-Counts* hinzuaddiert: In der 5. Zeile wird ein aktueller Satz aus dem Trainingskorpus aufgegriffen, für den in der 6. Zeile die Inside- und Outside-Wahrscheinlichkeiten berechnet werden. In der 7. bis 9. Zeile werden

dann damit in einer Schleife über alle Grammatikregeln die *(totalen) Grammatikregel- und Kategorie-Counts* berechnet und aktualisiert. In der 10. bis 11. Zeile wird für die aktuelle Trainingsiteration eine probabilistische kontextfreie Grammatik erstellt, indem die *totalen Grammatikregel-Counts* gemäss der Standard-Nebenbedingung für probabilistische kontextfreie Grammatiken (Booth und Thompson 1973) zu Grammatikregel-Wahrscheinlichkeiten normiert werden (siehe Abschnitt 3.9). In der 12. Zeile wird diese Grammatik ausgegeben.

Bemerkungen:

Da alle trainierten Grammatiken vom Inside-Outside-Algorithmus ausgegeben werden, ist es möglich (und beabsichtigt), dass extern (ausserhalb der Prozdeur) alle mit dieser Variation des Inside-Outside-Algorithmus trainierten Grammatiken linguistisch evaluiert werden. Dies ist Gegenstand des nächsten Abschnitts.

Eine Implementierung des Inside-Outside-Algorithmus wurde von Johnson (2001a) auf dem Web zur freien Verfügung gestellt. Leider wird dort eine feste Startgrammatik verwendet und die Anzahl der Iterationen wird über ein Konvergenzkriterium mit der Log-Likelihood bestimmt. Da diese beiden Parameter des Inside-Outside-Algorithmus aber einen wesentlichen Einfluss auf die Ergebnisse der linguistischen Evaluierung haben, müssen sie variiert werden, um optimale Ergebnisse zu erzielen. Da die Implementierung mit dem Source Code in ANSI C ausgeliefert wird, können entsprechende Änderungen an Johnsons Implementierung aber selbst vorgenommen werden.

Training und Evaluierung mit dem Inside-Outside-Algorithmus

Die Abbildung 3.6 zeigt das Training einer probabilistischen kontextfreien Grammatik mit dem Inside-Outside-Algorithmus, sowie die systematische Evaluierung seiner freien Parameter: Trainingskorpusgrösse, Anzahl Trainingsiterationen, Startgrammatik. In der Abbildung sind die Algorithmen mit dem Symbol ▭, die Daten mit dem Symbol ⛁ gekennzeichnet. Vergleicht man diese Abbildung mit der Abbildung 2.7, so fällt die deutlich höhere Komplexität des Trainings mit dem Inside-Outside-Algorithmus auf. Diese hohe Komplexität ist sicherlich dafür verantwortlich, dass das Baumbank-Trainingsverfahren einfacher durchzuführen ist als ein fehlerfreies Training mit dem Inside-Outside-Algorithmus. Möglicherweise erklärt sie auch, warum das Baumbank-Trainingsverfahren beliebter ist als der Inside-Outside-Algorithmus, bzw. warum in der Literatur oft von schlechten Ergebnissen des Inside-Outside-Algorithmus reportiert wird.

Die Abbildung 3.6 teilt sich, grob gesagt, in einen linken Teil (gestrichelt umrahmt) und einen rechten Teil.

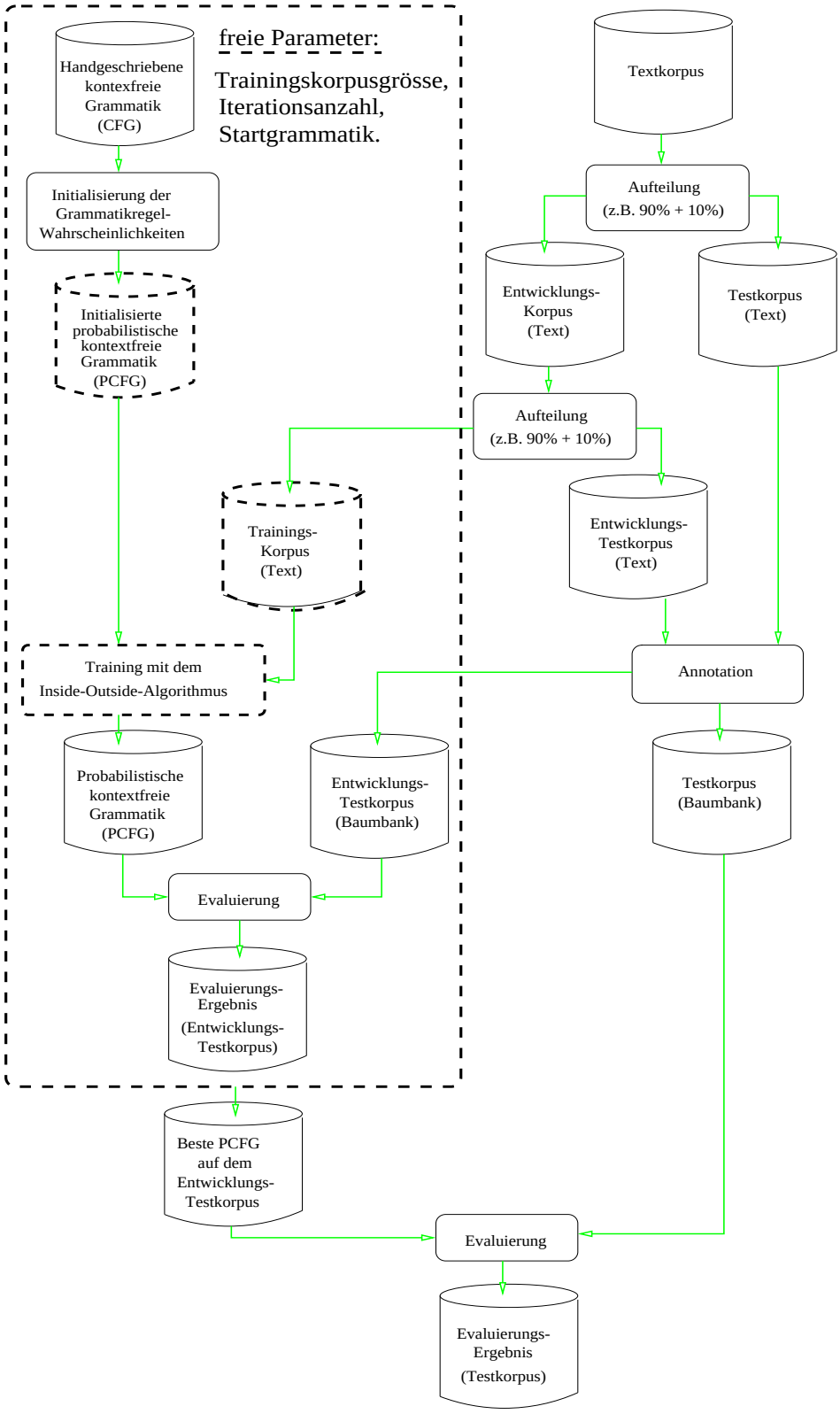


Abbildung 3.6: *Training* einer probabilistischen kontextfreien Grammatik mit dem *Inside-Outside-Algorithmus*, und *systematische Evaluierung* seiner *freien Parameter*: Trainingskorpusgrösse, Anzahl Trainingsiterationen, Startgrammatik.

Im rechten Teil der Abbildung wird die Aufteilung des gegebenen Textkorpus gezeigt, die in einen Trainingskorpus (Text), einen Entwicklungs-Testkorpus (in der Regel eine Baumbank), sowie einen Testkorpus (ebenfalls in der Regel eine Baumbank) resultiert. Die *Aufteilung* des Textkorpus in einen *Entwicklungskorpus* und einen *Testkorpus* bereitet den Test vor, wie gut die entwickelte Grammatik arbeitet. Um die Fähigkeit zur Generalisierung zu testen, sollten die Testdaten ungesehene Daten sein, d.h. Daten die nicht für die Entwicklung verwendet wurden. In der Regel wird die Aufteilung des gegebenen Textkorpus in Entwicklungs- und Testdaten so vorgenommen, dass man 90% Entwicklungsdaten und 10% Testdaten erhält [2]. Natürlich sollten die Testdaten unabhängig von den Entwicklungsdaten sein, aber diesen nicht zu unähnlich sein. Die übliche Prozedur, um dies zu erreichen, ist es, jeden zehnten Satz des gegebenen Textkorpus als Testdatum zu nehmen. Da während der Entwicklung mehrere Grammatiken anfallen, aber nur eine einzige Grammatik "entwickelt" werden soll, ist es nötig, dass der Entwicklungskorpus seinerseits aufgeteilt wird. Das Ergebnis dieser Aufteilung, die wie oben vorgenommen werden kann, indem jeder zehnte Satz des Entwicklungskorpus als Entwicklungs-Testdatum benutzt wird, sind 90% Trainingskorpus und 10% Entwicklungs-Testkorpus. Da es sich bei dem Testkorpus und dem Entwicklungs-Testkorpus um Textdaten handelt, müssen beide Testkorpora annotiert werden. Leider ist die manuelle Herstellung von annotierten Testkorpora sehr aufwendig. Es ist üblich, jeweils mindestens 300 - 600 Sätze zu annotieren. Je grösser diese Zahl ist, desto kleinere Auswirkungen werden gelegentliche Annotationsfehler haben, und desto robuster werden die Evaluierungs-Ergebnisse sein. Da die Annotation der Testdaten erhebliche Kosten verursacht, ist es sinnvoll, die Anzahl der Testdaten möglichst klein zu halten. Dies ist nicht ganz ungefährlich. In der Literatur liest man in diesem Zusammenhang immer wieder von "statistisch signifikanten" Evaluierungsergebnissen, die durch Anwendung der mathematischen Test-Theorie erhalten wurden. Leider müssen die meisten bekannten statistischen Tests annehmen, dass die getesteten Datentypen gewissen Wahrscheinlichkeitsverteilungen unterliegen. Natürlich ist es völlig unmöglich, zu entscheiden, ob diese Annahmen von den linguistischen Datentypen erfüllt werden. Daher sind die meisten Aussagen zur "statistischen Signifikanz" nahezu wertlos und geben höchstens Anhaltspunkte, ob das linguistische Evaluierungsergebnis möglicherweise robust sein könnte. Enthalten die linguistischen Evaluierungsergebnisse jedoch wenigstens die Angabe des Umfangs der Testdaten (oft genug fehlt diese Angabe), so kann sich jeder Leser zumindest seine eigene Meinung über die Robustheit der reportierten Evaluierungsergebnisse bilden. In der hervorragenden Arbeit von Yeh (2000) wird diese Problematik aufgegriffen und ein sehr einfaches und praktikables Evaluierungsverfahren für linguistische Daten vorge-

[2] Die 90:10-Regel ist auch eine bewährte Faustregel für überwachtes Training. Die tatsächliche Aufteilung hängt aber unter anderem auch davon ab, wieviele Daten annotiert werden können.

stellt, welches anstelle unkritisch und unpassend angewendeter statistischer Tests benutzt werden sollte.

Im linken Teil der Abbildung 3.6 wird der Inside-Outside-Algorithmus aus Abbildung 3.5 innerhalb einer *Schleife über alle freien Parameter* auf eine gegebene (in der Regel handgeschriebene) kontextfreie Grammatik angewendet. Die freien Parameter sind in der Abbildung gestrichelt dargestellt: es handelt sich um: (i) die *Startgrammatik*, d.h. die durch Zufallsinitialisierung der Grammatikregel-Wahrscheinlichkeiten hergestellten probabilistischen kontextfreien Grammatiken, (ii) den *Trainingskorpus (Text)*, sowie (iii) die *Anzahl der im Inside-Outside-Algorithmus vorzunehmenden Iterationen.* Naürlich sind Zahlenangaben zu der Anzahl der freien Parameter unseriös, da diese stets von der gegebenen kontextfreien Grammatik und einem eventuell gegebenem Zeitrahmen abhängen: Die Zahl der Startgrammatiken kann eigentlich nie hoch genug sein und sollte nur durch die Rechenzeit begrenzt werden, die man zur Verfügung hat, um ein Ergebnis vorzuweisen: 20 bis 100 Grammatiken sollten ausprobiert werden. Die Zahl der Iterationen ist einerseits unproblematisch, da gelegentlich schon nach zwei Iterationen die beste Grammatik erwischt wird. Da dies bewiesen werden muss, sollte man allerdings 5 bis 20 Iterationen vornehmen. Äusserst mangelhaft ist es allerdings, mit einer festen Anzahl von Iterationen zu arbeiten, da die Güte der trainierten Grammatiken in der Regel schnell erreicht wird, aber genauso schnell wieder sinkt. Arbeitet man also mit einer festen Iterationsanzahl, so arbeitet man fast ausschliesslich mit schlechten Grammatiken. Die Wichtigkeit der Grösse des Trainingskorpus wird wahrscheinlich überschätzt: man nimmt, was da ist und sinnvoll erscheint, und experimentiert mit $\frac{1}{10}\times$, $\frac{1}{100}\times$, $\frac{1}{1000}\times, \ldots$ Trainingskorpora. Für jede Kombination dieser freien Parameter wird der Inside-Outside-Algorithmus aus Abbildung 3.5 benutzt, um trainierte probabilistische kontextfreie Grammatiken zu berechnen. Jede der resultierenden Grammatiken wird mit Hilfe des Entwicklungs-Testkorpus evaluiert. Die Evaluierung einer Grammatik ist stets eine schwierige Aufgabe. Die Standard-Vorgehensweise ist, den Entwicklungskorpus mit Hilfe jeder trainierten Grammatik zu parsen und den jeweils wahrscheinlichsten Syntaxbaum (den Viterbi-Parse) mit dem im Testkorpus annotierten Syntaxbaum zu vergleichen. Hierfür stehen unter anderem die folgenden Evaluierungsmetriken zur Verfügung: Die PARSEVAL-Metriken: (Labeled) Precision, (Labeled) Recall und Crossing Brackets, sowie Akuratheit für Subkategorisierungsrahmen, Nomen-, Verb-, und Präpositional-Chunks, sowie Akuratheit für den Exact Match Task (siehe Abschnitt 2.5). Als Ausgabe der Schleife über die freien Parameter erhält man, nach der Evaluierung aller trainierten Grammatiken, die auf dem Entwicklungs-Testkorpus beste Grammatik. Diese Grammatik könnte man *die mit dem Inside-Outside-Algorithmus trainierte probabilistische Grammatik* nennen. Eine abschliessende Evaluierung dieser Grammatik auf dem Testkorpus liefert schliesslich das Evaluierungs-Ergebnis, welches reportiert werden darf.

3.2 Formale Grundlagen des EM-Algorithmus

Der *Erwartungswert-Maximierungs-Algorithmus* (EM-Algorithmus) wurde von Dempster et al. (1977) eingeführt und benannt. In diesem Artikel finden sich einige grundlegende Eigenschaften des EM-Algorithmus, sowie mehrere (nicht-linguistische) Beispiele und Anwendungen. Da die Idee hinter dem EM-Algorithmus so intuitiv und allgemein verständlich ist, wurde der EM-Algorithmus allerdings von vielen Autoren schon vor 1977 in einer Vielzahl von Anwendungen benutzt. Speziell für die Linguistik ist interessant, dass Baker (1979) den Inside-Outside-Algorithmus für kontextfreie Grammatiken in Chomsky-Normalform einführte und ihn mit dem Inside-Outside-Algorithmus für Hidden-Markov-Modelle verglich, welcher seinerseits von Baum (1972) in seiner endgültigen Form, als dynamische Programmiervariante, vorgestellt wurde. In Abschnitt 3.8 wird gezeigt werden, dass der Inside-Outside-Algorithmus eine Instanz des EM-Algorithmus für kontextfreie Grammatiken ist. Obwohl Baker seine Arbeit mit einer linguistischen Anwendung motivierte (Modellierung von Fehlern in der Spracherkennung), wird der EM-Algorithmus erst in neuerer Zeit in der Computer-Linguistik (insbesondere für Parsing und lexikalische Semantik) verstärkt angewendet.

In dieser Arbeit wird der EM-Algorithmus als stochastische Grundlage der *automatischen Lernverfahren* benutzt, die in Kapitel 4 und 5 vorgestellt werden. Die Ziele dieser Lernverfahren sind:

- semantische Klassifizierung von Verb-Argument-Daten,
- Induktion semantisch annotierter Lexika,
- Identifikation von Verb-Nomen-Kollokationen,
- Auflösung lexikalischer Ambiguitäten,
- phonologische Klassifizierung von Silben,
- Training von probabilistischen lexikalisierten kontext-freien Grammatiken,
- Training von stochastischen unifikationsbasierten Grammatiken.

Diese Lernverfahren zeichnen sich dadurch aus, dass es stochastische Verfahren sind, deren stochastischen Parameter in einer iterativen Trainingsphase vermittels des Maximum-Likelihood-Kriteriums optimal bestimmt werden. Ihr charakterisierendes Merkmal, was sie von den meisten anderen Lernverfahren unterscheidet, ist, dass dank der Benutzung des

EM-Algorithmuss in der Lernphase ein Trainingskorpus benutzt werden darf, der unannotiert (d.h. ambig) ist[3]. Das bedeutet:

- Zur semantischen Klassifizierung von Verb-Argument-Daten ist kein semantisch annotierter Trainingskorpus, wie SemCor, nötig,
- Die Induktion semantisch annotierter Lexika wird ohne Benutzung semantisch fixierter Taxonomien, wie WordNet oder GermaNet geschehen,
- Die Identifikation von Verb-Nomen-Kolloktionen ist auf der Grundlage automatisch klassifizierter Verb-Nomen-Paare möglich,
- Die Auflösung lexikalischer Ambiguitäten gelingt ohne Zugriff auf WordNet- bzw. Hector- Annotationen,
- Eine phonologische Klassifizierung von Silben ist auf der Grundlage grosser transkribierter Texte möglich,
- Ein unbeaufsichtigtes Training lexikalisierter probabilistischer kontext-freier Grammatiken ist ohne Baumbanken möglich, wie sie beispielsweise für Englisch mit der Penn-Treebank, oder dem Negra-Korpus für Deutsch, in relativ kleinem Umfang zur Verfügung stehen,
- Ein Training lexikalisierter stochastischer unifikationsbasierter Grammatiken ist ohne Parse-Banken, die nur in einem kleinen Umfang vorhanden sind, z.B. der Homecenter-Korpus oder Verbmobil-Baumbank, nicht nur möglich, sondern zeigt auch sehr viel bessere Ergebnisse.

Neben den schon genannten Artikeln von Baum und Eagon (1966), Baum und Sell (1968), Baum et al. (1970), Baum (1972), Dempster et al. (1977), Baker (1979) ist das Buch von McLachlan und Krishnan (1997) eine empfehlenswerte Literaturquelle zum EM-Algorithmus. Leider erschliessen sich diese Arbeiten ohne mathematische Kenntnisse nur schwer.

Formale Definition des EM-Algorithmus

Rein formal kann der EM-Algorithmus als ein 8-Tupel:

$$< \mathcal{Y},\ \tilde{p}(.),\ \mathcal{X},\ X(.),\ \Omega,\ \Omega_0,\ \{p_\Theta(.)|\Theta \in \Omega\},\ \mathcal{M}(.) >$$

[3]Ein Standardwerk für maschinelle Lernverfahren ist Mitchell (1997). Der EM-Algorithmus wird allerdings erst in Kapitel 6.12 behandelt, während einige seiner einfachen Spezialfälle, z.B. die auf *Entscheidungsbäumen* basierenden Lernverfahren, bevorzugt präsentiert werden. Ein weiteres interessantes Buch ist Frey (1998), in dem "graphische Modelle" präsentiert werden, die eine Reihe von stochastischen Lernverfahren verallgemeinern.

mit den folgenden Elementen definiert werden:

- $\mathcal{Y}$, die Menge der *unvollständigen Datentypen*,
- $\tilde{p}(.)$, die *empirische Wahrscheinlichkeitsverteilung der unvollständigen Datentypen*,
- $\mathcal{X}$, die Menge der *vollständigen Datentypen*,
- $X(.)$, die *symbolische Analyse-Komponente*,
- Ω, der *Parameterraum*,
- Ω_0, die *Menge der Startparameter*,
- $\{p_\Theta(.) \mid \Theta \in \Omega\}$, die Menge der *parametrisierten vollständigen Wahrscheinlichkeitsverteilungen*,
- $\mathcal{M}(.)$, die *EM-Parameter-Abbildung*.

Für die Anwendung des EM-Algorithmus müssen diese acht Elemente spezifiziert, d.h. berechnet oder mit Werten initialisiert werden. Im folgenden werden sie definiert.

- Die Menge $\mathcal{Y}$ der *unvollständigen Datentypen* ist eine diskrete (endliche oder abzählbar unendliche) Menge von linguistischen Datentypen. Dem Sprachgebrauch liegt die Vorstellung zugrunde, dass diese Datentypen zusätzliche strukturelle Eigenschaften besitzen, die versteckt sind, aber mit geeigneten Analysemethoden aufgedeckt werden können. Daher ist es angemessen, sich unvollständige Datentypen als im Prinzip linguistisch reich strukturierbare Daten vorzustellen, von denen aber lediglich die "Oberfläche" sichtbar ist. Ein geeignetes Beispiel unvollständiger Datentypen sind die natürlichsprachigen Sätze (Wortfolgen). Im Rahmen des EM-Algorithmus werden sie als unvollständig angesehen, wenn sie (was in freiem Text, nicht aber in einer Baumbank, der Fall ist) ohne die zusätzliche Angabe ihrer grammatischen Analysen vorliegen.
- Die *empirische Wahrscheinlichkeitsverteilung* $\tilde{p}(.)$ *der unvollständigen Datentypen* ist wie in Abschnitt 2.1 definiert. Jedem unvollständigen Datentyp $y \in \mathcal{Y}$ wird als Wahrscheinlichkeit die relative Häufigkeit $f(y)$ seines Vorkommens in einem gegebenen Korpus zugewiesen:

  $$\tilde{p} : \mathcal{Y} \to [0,1], \quad y \mapsto |f|^{-1} \cdot f(y) \ .$$

  Hierbei ist $|f| = \sum_{y \in \mathcal{Y}} f(y)$ die Grösse des Korpus, ausgedrückt durch die Anzahl aller seiner Datentoken. Die Korpusgrösse muss daher als ein Parameter des EM-Algorithmus angesehen werden, der möglicherweise (vermittels der empirischen Wahrscheinlichkeitsverteilung) die Resultate des EM-Algorithmus beeinflussen wird.

- Die Menge $\mathcal{X}$ der *vollständigen Datentypen* ist ebenfalls eine diskrete Menge von linguistischen Datentypen. Im Vergleich mit den unvollständigen Datentypen sollte man sich die vollständigen Datentypen als linguistisch reich strukturierte Datentypen vorstellen, die keinerlei relevante Information vermissen lassen und die Informationen der unvollständigen Datentypen ergänzen. Ein geeignetes Beispiel vollständiger Datentypen sind die linguistischen Analysen von natürlichsprachigen Sätzen, die als Datentoken etwa in Baumbanken vorliegen. Im Rahmen des EM-Algorithmus können diese linguistische Analysen (Bäume) als vollständig angesehen werden, weil sie Sätze (Wortfolgen) um strukturelle, linguistische Informationen anreichern. Natürlich ist es in diesem Zusammenhang völlig unerheblich, ob man für die Beschreibung der linguistischen Informationen einen kontextfreien oder unifikationsbasierten Formalismus wählt.

- Die *symbolische Analyse-Komponente* $X(.)$ hat die Fähigkeit, jeden unvollständigen Datentyp analysieren zu können, d.h. dessen versteckte, strukturelle Eigenschaften mit geeigneten Methoden aufzudecken. Da ein unvollständiger Datentyp $y \in \mathcal{Y}$ mehr als eine einzige Analyse besitzen darf, ist es zweckmässig, die Menge $X(y)$ aller seiner Analysen als eine Teilmenge der vollständigen Datentypen zu formalisieren. Folglich ist die symbolische Analyse-Komponente $X(.)$ eine Abbildung von der Menge der unvollständigen Datentypen auf die Potenzmenge der vollständigen Datentypen:

$$X : \mathcal{Y} \to \mathcal{P}(\mathcal{X}), \quad y \mapsto X(y) \ .$$

  Das typische Beispiel einer symbolischen Analyse-Komponente ist der symbolische Parser, da dieser einem Satz (d.h. einem unvollständigen Datentyp) unter Benutzung einer geeigneten Grammatik eine Menge von Satzanalysen (d.h. eine Teilmenge der vollständigen Datentypen) zuweisen wird. Weitere Beispiele symbolischer Analyse--Komponenten sind semantische Tagger, aber natürlich auch gewöhnliche Part-Of-Speech-Tagger. Die natürlichsprachigen Wörter können in diesem Zusammenhang als unvollständige, und die Wort-Tag-Paare als vollständige Datentypen angesehen werden. Ein symbolischer Tagger bildet dann jedes Wort auf eine Anzahl von möglichen Wort-Tag-Paaren ab.

  Es ist überaus wichtig, dass die symbolische Analyse-Komponente im Szenario des EM-Algorithmus die folgenden zwei Eigenschaften besitzt: (i) kein vollständiger Datentyp darf die Analyse von zwei verschiedenen unvollständigen Datentypen sein, (ii) jeder vollständige Datentyp ist die Analyse von wenigstens einem unvollständigen Datentyp. Diese Eigenschaften könnten mit *Disjunkheit* und *Vollständigkeit*

umschrieben werden. Formal gilt:

$$\begin{array}{ll} (\textit{Disjunkheit}) & X(y) \cap X(y') = \emptyset \quad \text{für } y \neq y' \ , \\ (\textit{Vollständigkeit}) & \bigcup_{y \in \mathcal{Y}} X(y) \supseteq \mathcal{X} \quad . \end{array}$$

Natürlich führen Disjunkheit und Vollständigkeit dazu, dass die Menge der vollständigen Datentypen in die Menge der Analysen der unvollständigen Datentypen *partitioniert* wird:

$$\sum_{y \in \mathcal{Y}} X(y) = \mathcal{X} \ .$$

Sobald die für den EM-Algorithmus relevanten Wahrscheinlichkeitsverteilungen eingeführt werden, wird klar werden, warum für die Analysen Disjunkheit und Vollständigkeit gefordert wird. An dieser Stelle soll zumindest darauf hingewiesen werden, dass die bisher eingeführten zwei Beispiele von symbolischen Analyse-Komponente diese beiden Eigenschaften besitzen. Ein symbolischer Syntax-Parser wird keinen zwei verschiedenen Sätzen dieselbe Analyse zuweisen, auch wird er jedem grammatischen Satz analysieren können (wenn schon nicht in der Praxis, dann zumindest in der Theorie). Ein (semantischer oder Part-Of-Speech) Tagger wird zwei verschiedenen Worten zwar möglicherweise dasselbe Tag zuweisen, aber nicht dasselbe Wort-Tag-Paar. Aus diesem Grund ist es sehr wichtig, dass nicht die Tags, sondern die Wort-Tag-Paare als vollständige Daten benutzt werden[4]. Ferner wird ein EM-basierter Tagger kein Wort unanalysiert lassen.

- Der *Parameterraum* Ω ist im allgemeinen eine kontinuierliche Menge von reellwertigen, mehr-dimensionalen Veränderlichen, die einer Reihe von Nebenbedingungen unterworfen sein können. Parameter $\Theta \in \Omega$ dienen in der Regel einem einfachen Zweck: sie sollen relevante Entitäten, in dieser Arbeit sind dies Wahrscheinlichkeitsverteilungen, vermittels simpler Indizierung benennen und zu einer weiterführenden Betrachtung freigeben. Ein Parameterraum muss im Rahmen des EM-Algorithmus eine Reihe mathematischer Eigenschaften aufweisen, deren Zusammenspiel aber zum grössten Teil noch unerforscht ist. In dieser Arbeit wird versucht, an geeigneter Stelle auf einige dieser Eigenschaften hinzuweisen.

- Die *Menge der Startparameter* Ω_0 ist eine endliche, in der Regel relativ kleine Teilmenge des Parameterraumes:

$$\Omega_0 \subseteq \Omega \ .$$

Im Gegensatz zu vielen Darstellungen des EM-Algorithmus, die mit einem einzigen Startparameter arbeiten, wird in dieser Arbeit eine leicht modifizierte Version des

[4]Die Tags repräsentieren in diesem Fall in sehr anschaulicher Weise die Information, die aus einem unvollständigen Datentyp (Wort) einen vollständigen Datentyp (Wort-Tag-Paar) macht.

EM-Algorithmus vorgestellt, die es nicht nur erlaubt, sondern sogar explizit fördert, mit einer beliebigen Anzahl von Startparametern zu arbeiten. Es wird vermutet, dass die hier vorgestellte Version schon für $|\Omega_0| = 2$ in den meisten Fällen wesentlich bessere Ergebnisse liefern wird, als für $|\Omega_0| = 1$ (Standard-Version in der Literatur). Das hat den folgenden Grund: Der EM-Algorithmus ist ein iterativer Algorithmus, von dem theoretisch gezeigt werden kann, dass er zu jedem fixen Startparameter $\Theta_0 \in \Omega$ eine Folge von (in einem bestimmten Sinn) immer besser werdenden Parametern produziert. Es ist sehr gut bekannt, dass verschiedene Startparameter in der Praxis oft zu qualitativ sehr verschiedenen Ergebnissen führen können (Müller (2000), Müller (2001b)). Leider weiss man aber nur in einigen wenigen Spezialfällen (Riezler et al. 2000), welche Wahl des Startparameters Θ_0 (zumindest theoretisch) optimal sein könnte. Konzeptionell ist es daher sinnvoll, den EM-Algorithmus nicht nur mit einem einzigen Startparameter durchzuführen, sondern mit mehreren Startparametern $\Theta_0 \in \Omega_0$.

- Die Menge $\{p_\Theta(.) \mid \Theta \in \Omega\}$ fasst die *parametrisierten Wahrscheinlichkeitsverteilungen* $p_\Theta(.)$ *über der Menge* $\mathcal{X}$ *der vollständigen Datentypen*, auch *parametrisierte vollständige Wahrscheinlichkeitsverteilungen* genannt, für alle Parameter $\Theta \in \Omega$ des Parameterraumes zusammmen. Es gilt also:

$$p_\Theta : \mathcal{X} \to [0,1], \quad x \mapsto p_\Theta(x) \ ,$$

mit:

$$p_\Theta(\mathcal{X}) = \sum_{x \in \mathcal{X}} p_\Theta(x) = 1 \ .$$

Im Rahmen des EM-Algorithmus sind die parametrisierten vollständigen Wahrscheinlichkeitsverteilungen diejenigen Objekte, auf die sich das ganze weitere Interesse bündelt. Dies kann in zwei Schritten eingesehen werden: (i) die parametrisierten vollständigen Wahrscheinlichkeitsverteilungen können (siehe unten) zu parametrisierten unvollständigen Wahrscheinlichkeitsverteilungen verallgemeinert werden, (ii) auf der Grundlage der unvollständigen Wahrscheinlichkeitsverteilungen und dem gegebenen Korpus $\tilde{p}(.)$ von unvollständigen Daten könnte mit der Maximum-Likelihood-Estimierung ein optimaler Parameter bestimmt werden. Die Maximum-Likelihood-Estimierung zur Bestimmung eines Parameters, welcher dem gegebenen Korpus $\tilde{p}(.)$ eine möglichst hohe Wahrscheinlichkeit zuweist, wurde bereits in Abschnitt 2.4 motiviert. An dieser Stelle soll daher auf Schritt (i) eingegangen werden. Es ist beachtenswert, dass die parametriserten unvollständigen Wahrscheinlichkeitsverteilungen in sehr natürlicher Weise zu Wahrscheinlichkeitsverteilungen über der

Menge der unvollständigen Daten führen, die im folgenden *parametrisierte unvollständige Wahrscheinlichkeitsverteilungen* genannt werden:

$$p_\Theta : \mathcal{Y} \to [0,1], \quad y \mapsto p_\Theta(X(y)) \ .$$

Wie bereits in dieser Definition angegeben wurde, ist es hierzu lediglich nötig, jeden unvollständigen Datentyp $y \in \mathcal{Y}$ mit der Menge seiner Analysen $X(y)$ zu identifizieren, d.h. die Wahrscheinlichkeit dieses unvollständigen Datentyps als die Wahrscheinlichkeit des Ereignisses $X(y)$ zu definieren:

$$p_\Theta(y) := p_\Theta(X(y)) \ .$$

Gemäss Abschnitt 2.2 ist dies die Wahrscheinlichkeit, dass mindestens eine der Analysen $x \in X(y)$ eintritt:

$$p_\Theta(X(y)) = \sum_{x \in X(y)} p_\Theta(x) \ .$$

Es kann nun leicht überprüft werden, dass aus dieser Definition tatsächlich eine Wahrscheinlichkeitsverteilung auf den unvollständigen Datentypen resultiert. Hierzu benutzt man als Schlüsselschritt, dass die Menge der vollständigen Datentypen in die Menge der Analysen der unvollständigen Datentypen partitioniert wird (siehe oben):

$$p_\Theta(\mathcal{Y}) = \sum_{y \in \mathcal{Y}} p_\Theta(y) = \sum_{y \in \mathcal{Y}} p_\Theta(X(y)) = p_\Theta\left(\sum_{y \in \mathcal{Y}} X(y)\right) = p_\Theta(\mathcal{X}) = 1 \ .$$

- Aus prozeduraler Sicht besteht der EM-Algorithmus in einer iterativen Anwendung der *EM-Parameter-Abbildung* $\mathcal{M}(.)$ auf die Startwerte des EM-Algorithmus. Somit könnte die EM-Parameter-Abbildung als der eigentliche Kern des EM-Algorithmus bezeichnet werden. Formal ist sie eine Abbildung von dem Parameterraum auf sich selbst und legt zu jedem Parameter $\Phi \in \Omega$ in zwei Schritten, (i) dem *Erwartungswert-Schritt (E-Schritt)* und (ii) dem *Maximierungs-Schritt (M-Schritt)* einen *re-estimierten Parameter* $\hat{\Theta} \in \Omega$ fest. Offensichtlich waren beide Schritte für den EM-Algorithmus namensgebend. Von der folgenden Definition der EM-Parameter-Abbildung lässt sich der M-Schritt bereits deutlich ablesen, der in der Maximierung der sogenannten EM-Hilfsfunktion besteht:

$$\mathcal{M} : \Omega \to \Omega, \quad \Phi \mapsto \hat{\Theta} = \text{argmax}_{\Theta \in \Omega} \ Q_\Phi(\Theta) \ .$$

Der dem M-Schritt vorangehende E-Schritt besteht darin, die *EM-Hilfsfunktion* $Q_\Phi(.)$ zu berechnen. Die EM-Hilfsfunktion ordnet jedem Parameter einen Wert zu,

von dem gezeigt werden wird, dass er (i) ein doppelter Erwartungswert ist, und (ii) in enger Verwandschaft mit der Log-Likelihood steht:

$$Q_\Phi : \Omega \to \mathbb{R}, \quad \Theta \mapsto \sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in X(y)} p_\Phi(x|y) \cdot \log\ p_\Theta(x) \ .$$

Die in dieser Definition vorkommende *bedingte Wahrscheinlichkeit* $p_\Phi(x|y)$ *einer Analyse* $x \in X(y)$ *gegeben deren unvollständigen Datentyps* $y \in \mathcal{Y}$ kann aus der gewöhnlichen Definition der bedingten Wahrscheinlichkeit abgeleitet werden. Hierzu ist es wieder nur nötig, wie bei der Definition der unvollständigen Wahrscheinlichkeitsverteilungen, einen unvollständigen Datentyp y mit seinen Analysen $X(y)$ zu identifizieren:

$$p_\Phi(x|y) = p_\Phi(x|X(y)) \ .$$

Am Ende von Abschnitt 2.2 wurden bedingte Wahrscheinlichkeiten formal eingeführt. Unter Benutzung dieser Definition folgt:

$$p_\Phi(x|X(y)) = \frac{p_\Phi(x)}{p_\Phi(X(y))} = \frac{p_\Phi(x)}{\sum_{x \in X(y)} p_\Phi(x)} \ .$$

Der EM-Algorithmus ist mit den vorangehenden Ausführungen formal völlig ausreichend erklärt worden. Da es aber für die Namensgebung des EM-Algorithmus massgebend ist, soll noch darauf hingewiesen werden, dass die Werte $Q_\Phi(\Theta)$ der EM-Hilfsfunktion als (doppelte) Erwartungswerte aufgefasst werden können. Um die Definition der Erwartungswerte anwenden zu können, ist es nötig, dass die bedingten Wahrscheinlichkeiten $p_\Phi(x|y)$ für jeden unvollständigen Datentyp $y \in \mathcal{Y}$ eine Wahrscheinlichkeitsverteilung $p_\Phi(.|y)$ auf der Menge seiner Analysen $x \in X(y)$ bilden:

$$p_\Phi(.|y) : X(y) \to [0,1], \quad x \mapsto \frac{p_\Phi(x)}{\sum_{x \in X(y)} p_\Phi(x)} \ ,$$

mit:

$$\sum_{x \in X(y)} p_\Phi(x|y) = 1 \ .$$

Die letzte Gleichung ist sehr einfach nachzuprüfen und erlaubt es, die EM-Hilfsfunktion als eine doppelte Erwartung zu identifizieren. Unter Benutzung der Definition der Erwartungswerte (siehe Abschnitt 2.3) gilt:

$$\begin{aligned} Q_\Phi(\Theta) &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in X(y)} p_\Phi(x|y) \cdot \log\ p_\Theta(x) \\ &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \cdot p_\Phi(.|y) \left[\ \log\ p_\Theta \ \right] \\ &= \tilde{p} \left[\ p_\Phi(.|.) \left[\ \log\ p_\Theta \ \right] \right] \ . \end{aligned}$$

Zusammenfassend folgt, dass ein re-estimierter Parameter im EM-Algorithmus (iterativ) wie folgt berechnet wird:

$$\begin{aligned} \hat{\Theta} &= \text{argmax}_{\Theta \in \Omega} \; Q_{\Phi}(\Theta) \\ &= \text{argmax}_{\Theta \in \Omega} \; \tilde{p}\,[\, p_{\Phi}(.|.)\,[\, \log \; p_{\Theta} \,]\,] \;. \end{aligned}$$

Prozedurale Sicht auf den EM-Algorithmus

Abbildung 3.7 zeigt den EM-Algorithmus aus prozeduraler Sicht. Seine Eingabe besteht aus einem Trainingskorpus $\tilde{p}(.)$ von unvollständigen Datentypen, einer symbolischen Analyse-Komponente $X(.)$, die zu jedem unvollständigen Datentyp $y \in \mathcal{Y}$ die Menge $X(y)$ seiner Analysen bereitstellt. Als weitere Eingabe liegen parametrisierte vollständige Wahrscheinlichkeitsverteilungen $p_{\Theta}(.)$ vor, für die (in einem gewissen Sinn) optimale Parameter zu bestimmen sind. Ferner gehört zu dieser Version des EM-Algorithmus im Gegensatz zur Standardversion nicht nur ein einziger, sondern eine Menge $\Omega_0 \subseteq \Omega$ von Startparametern, sowie anstatt eines Abruchkriteriums eine (im allgemeinen relativ hohe) Anzahl $numberOfIterations \geq 1$ von festgelegten Iterationen. Die Bevorzugung vieler Startparameter gegenüber einem einzigen Startparameter wurde schon motiviert. Die Benutzung einer hohen Anzahl von Iterationen ist gegenüber einem Log-Likelihood-basierten Abruchkriterium zu bevorzugen, weil ein (für eine konkrete linguistische Anwendung) optimaler Parameter nur mit einer zusätzlichen linguistischen Evaluierung bestimmt werden kann, und diese linguistische Evaluierung erfahrungsgemäss selten mit einem Log-Likelihood-basierten Abruchkriterium korreliert.

Die Ausgabe des EM-Algorithmus besteht aus einer Folge von re-estimiertemn Parametern

$$\Theta_1, \; \Theta_2, \; \Theta_3, \; \ldots \in \Omega \;,$$

d.h. aus Parametern, die durch eine iterierte Anwendung der EM-Parameter-Abbildung $\mathcal{M}(.)$ enstehen:

$$\Theta_i = \mathcal{M}(\Theta_{i-1}) \qquad (i = 1, 2, 3, \ldots) \;.$$

Dempster et al. (1977) zeigten, dass die zu diesen Parametern gehörige Folge der Log-Likelihood-Werte auf dem Trainingskorpus beschränkt und monoton wachsend ist:

$$L(\Theta_1) \leq L(\Theta_2) \leq L(\Theta_3) \leq \ldots \in [L(\Theta_0), \; 0] \;.$$

Während ein einfacher Satz der Analysis besagt, dass jede beschränkte und monoton wachsende Folge *konvergent* ist, d.h.

$$L(\Theta_i) \to L^* \qquad (i \to \infty) \;,$$

Eingabe: Ein *Trainingskorpus* $\tilde{p}(.)$ *von unvollständigen Datentypen*, eine *symbolische Analyse-Komponente* $X(.)$, parametrisierte vollständige Wahrscheinlichkeitsverteilungen $p_\Theta(.)$, eine endliche Menge Ω_0 von *Startparametern*, sowie eine *Iterationsanzahl* $numberOfIterations \geq 1$.

Ausgabe: Für jeden Startwert $\Theta_0 \in \Omega_0$ eine Folge von re-estimierten Parametern:

$$\Theta_1,\ \Theta_2,\ \Theta_3,\ \ldots \in \Omega\ ,$$

sodass die zugehörige Folge der Log-Likelihood-Werte auf dem Trainingskorpus beschränkt und monoton wachsend ist:

$$L(\Theta_1) \leq L(\Theta_2) \leq L(\Theta_3) \leq \ldots \in [L(\Theta_0),\ 0]\ .$$

Prozedur:

1. * Version mit Variation der Startparameter: *\
2. **for each** $\Theta_0 \in \Omega$ **do**
3. **for each** $i := 1, \ldots,$ *numberOfIterations* **do**
4. $\Phi := \Theta_{i-1}$;
5. **(E-step) compute** (as function of Θ)

$$Q_\Phi(\Theta) = \sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in X(y)} p_\Phi(x|y) \cdot \log\ p_\Theta(x);$$

6. **(M-step) compute**

$$\hat{\Theta} = \text{argmax}_{\Theta \in \Omega}\ \ Q_\Phi(\Theta);$$

7. $\Theta_i := \hat{\Theta}$;
8. **print** $\Theta_0,\ \Theta_1,\ \Theta_2,\ \Theta_3,\ \ldots$;

Abbildung 3.7: EM-Algorithmus

konnte Wu (1983) (mit zusätzlichen Voraussetzungen an den EM-Algorithmus) zwei weitere schöne Eigenschaften des EM-Algorithmus zeigen: (i) L^* ist ein lokales Maximum der Log-Likelihood und (ii) die Folge der Parameter konvergiert:

$$\Theta_i \to \Theta^* \qquad (i \to \infty) \ .$$

Beide Eigenschaften sind natürlich sehr wünschenswert, da sie es ermöglichen, den EM-Algorithmus als ein iteratives Lösungsverfahren für die in Kapitel 2 vorgestellte Maximum-Likelihood-Estimierung aufzufassen.

In der 2. Zeile der Prozedur des EM-Algorithmus wird eine Schleife über die Startparameter initialisiert, deren Schritte in der 3. bis 8. Zeile ausgeführt werden. In der 3. Zeile wird eine Schleife über die Trainingsiterationen initialisiert, deren Schritte in der 4. bis 7. Zeile ausgeführt werden. In der 4. Zeile wird der sogenannte *gegenwärtige Parameter* Φ der EM-Hilfsfunktion gesetzt, die als E-Schritt in der 5. Zeile berechnet wird. In der 6. Zeile wird der M-Schritt ausgeführt, der die EM-Hilfsfunktion maximiert und den zugehörigen sogenannten *neuen Parameter* $\hat{\Theta}$ bestimmt. Ohne eine konkrete Instantiierung des EM-Algorithmus stellen die 5. und 6. Zeile die allgemeinste Vorgehensweise für die Berechnung des neuen Parameters $\hat{\Theta}$ dar. Es wurde schon von Dempster et al. (1977) darauf hingewiesen, dass der Ausdruck "Algorithmus" für den EM-Algorithmus kritisiert werden könnte, weil die Sequenz der Berechnungsschritte, die für die Ausführung eines E- oder M-Schrittes nötig sind, nicht spezifiziert werden kann. Es ist offensichtlich, dass detaillierte Implementationen des EM-Algorithmus sich hinsichtlich Komplexität und Anwendbarkeit stark voneinander unterscheiden können. In einer konkreten Instanz des EM-Algorithmus ist es oft angebracht, beide Schritte zusammenzufassen, um eine konkrete mathematische Lösung für $\hat{\Theta}$ auszuarbeiten. In der Regel ist eine solche Lösung relativ einfach zu finden (siehe z.B. Abschnitt 3.8). Manchmal jedoch, wie z.B. bei den *log-linearen Modellen* (siehe Abschnitt 5.3), kann eine Lösung nur in einem iterativen Verfahren gefunden werden, sodass der EM-Algorithmus insgesamt zu einem sogenannten *doppelt-iterativen Algorithmus* wird (wenn die Schleife über die Startparameter nicht mitgezählt wird, wie es beim Standard-EM-Algorithmus leider üblich ist). In der 7. Zeile wird der neue neue Parameter gespeichert, um ihn für die nächste Iteration (als gegenwärtigen Parameter) benutzen zu können. In der 8. Zeile wird zu jedem Startparameter Θ_0 die Folge der aus ihm berechneten re-estimierten Parameter ausgegeben.

Bemerkung: Da alle re-estimierten Parameter vom EM-Algorithmus ausgegeben werden, ist es sowohl sinnvoll, als auch leicht möglich, eine zusätzliche systematische linguistische Evaluierung aller vom EM-Algorithmus berechneten Parameter extern vorzunehmen.

3.3 Anwendung und Interpretation des EM-Algorithmus

In diesem Abschnitt werden zwei Ziele verfolgt:

1. In Abschnitt 3.2 wurde der EM-Algorithmus formal eingeführt. Um ein Gefühl für mögliche Anwendungsumgebungen des EM-Algorithmus zu vermitteln, sollen die dafür wesentlichen Elemente des EM-Algorithmus ein weiteres Mal vorgestellt werden. Hierbei wird die Methodik verfolgt, zuerst die vertrauten *symbolischen Komponenten* des EM-Algorithmus vorzustellen, und dann zu den vielleicht weniger vertrauten *stochastischen Basiskomponenten*, sowie den aus ihnen *abgeleiteten stochastischen Verteilungen und objektiven Funktionen* überzugehen.

2. Die stochastischen Komponenten des EM-Algorithmus, insbesondere aber die nützlichste stochastische Funktion des EM-Algorithmus, nämlich die EM-Hilfsfunktion, sollen interpretiert werden. Als Schlüsselschritt wird sich hierbei die Einführung der sogenannten *empirische Wahrscheinlichkeitsverteilung der vollständigen Datentypen* herausstellen. Mit dieser Wahrscheinlichkeitsverteilung über vollständigen Datentypen wird es gelingen die EM-Hilfsfunktion als eine *Log-Likelihood auf vollständigen Datentypen* zu interpretieren, sodass der EM-Algorithmus als eine iterative Maximum-Likelihood-Estimierung von schrittweise verbesserten Log-Likelihood-Funktionen interpretierbar ist. (Obwohl diese Interpretation sehr befriedigend ist, beantwortet sie zunächst noch nicht die Frage, ob das Ergebnis der iterativen Maximum-Likelihood-Estimierungen etwas mit der Maximum-Likelihood-Estimierung der Log-Likelihood des EM-Algorithmus zu tun hat. Dieser Komplex wird ausführlich in Abschnitt 3.6 behandelt werden.)

Anwendung des EM-Algorithmus

Der EM-Algorithmus besteht aus drei *symbolischen Komponenten*, zwei *stochastischen Basiskomponenten*, sowie einigen *abgeleiteten stochastischen Wahrscheinlichkeitsverteilungen und objektiven Funktionen*. Die drei symbolischen Komponenten des EM-Algorithmus, die in Abbildung 3.8 gezeigt werden, gehören zu einer Anwendungsumgebung des EM-Algorithmus, in der diese spezifiziert sein müssen. Es handelt sich um:

- die unvollständigen Datentypen $\mathcal{Y}$,
- die vollständigen Datentypen $\mathcal{X}$,
- sowie die *symbolische Analyse-Komponente* $X(.)$, die jeden unvollständigen Datentyp $y \in \mathcal{Y}$ *analysiert* und ihm als Ergebnis eine Menge $X(y) \subseteq \mathcal{X}$ von vollständigen

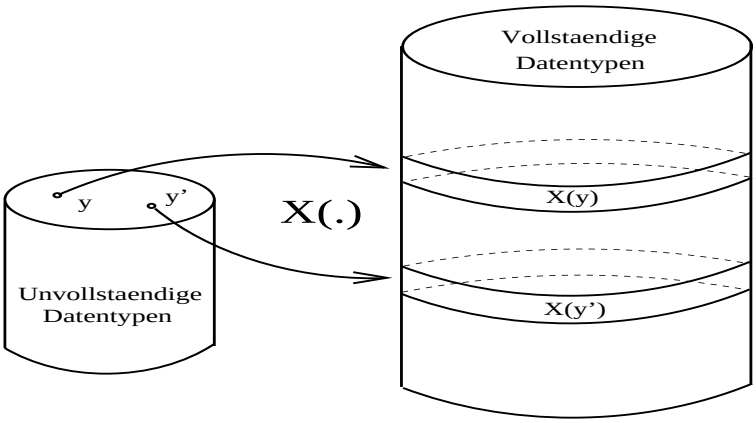


Abbildung 3.8: Symbolische Komponenten des EM-Algorithmus

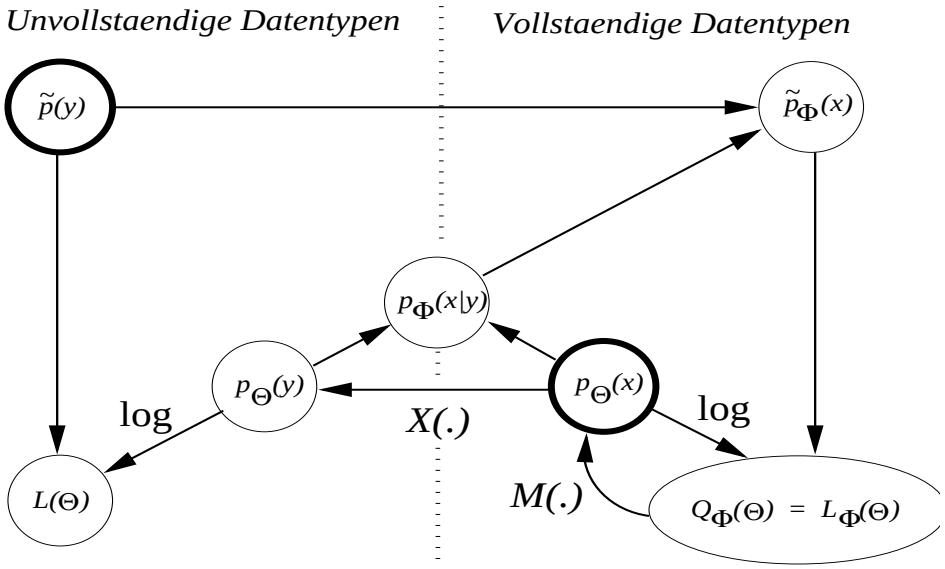


Abbildung 3.9: Stochastische Komponenten des EM-Algorithmus

Datentypen zuweist.

Die Abbildung 3.8 zeigt, dass die symbolische Analyse-Komponente (i) einem unvollständigen Datentyp eine Menge vollständiger Datentypen zuweist, und (ii) verschiedene unvollständige Datentypen $y \neq y'$ auf disjunkte *Analysen* $X(y) \cap X(y') = \emptyset$ abbbildet.

Der symbolische Ansatz des EM-Algorithmus ist daher so allgemein, dass er auf eine Vielzahl von linguistischen Analyse- und Generierungswerkzeugen, wie Speech-Recognizer, Tagger, Silbifizierer, Morphological-Analyzer, Chunker, Parser (mit und ohne Semantikanalyse), Speech-Synthesizer, Generierer, Graphem-Phonem-Systeme, Text-To-Speech-Systeme, Übersetzungs-Systeme, u.s.w. zutreffen wird.

Abbildung 3.9 zeigt, dass der EM-Algorithmus eine Vielzahl von stochastischen Komponenten besitzt, aber nur zwei (**fett umkreiste**) stochastische Basiskomponenten:

- den durch die *empirische Wahrscheinlichkeitsverteilung* $\tilde{p}(y),\ y \in \mathcal{Y}$, gegebenen *Korpus von unvollständigen Datentypen*, sowie
- die *parametrisierte vollständige Wahrscheinlichkeitsverteilung* $p_\Theta(x),\ x \in \mathcal{X}$, welche gelegentlich auch als *Wahrscheinlichkeitsmodell* bezeichnet wird. Implizit sind mit der parametrisierten vollständigen Wahrscheinlichkeitsverteilung natürlich auch die Parameter $\Theta \in \Omega$ gegeben, sowie die zufälligen Startparameter $\Omega_0 \subseteq \Omega$.

Der Sinn ist der folgende: selbst bei der Entwicklung *symbolischer Analysewerkzeuge* müssen im allgemeinen gewisse *Heuristiken* benutzt werden, um dem Werkzeug, in Situationen in denen es vor der Wahl zwischen mehreren möglichen Alternativen der Problembehandlung steht, die Fähigkeit der Entscheidungsfindung zu verleihen. Im allgemeinen manifestieren sich Heuristiken durch die Belegung gewisser Werkzeug-interner Variablen mit bestimmten Werten, die durch Erfahrung und Intuition diktiert werden. Damit stark kontrastierend, wird bei der Entwicklung eines stochastischen Analysewerkzeugs zwar nicht auf Heuristiken und Parameter verzichtet, wohl aber auf die weitgehend unmotivierte Festlegung von Parameterwerten. Ein *stochastisches Analysewerkzeug*, wie es z.B. im EM-Algorithmus vorliegt, kann als ein Paar

$$< X(.),\ p_\Theta(.) >$$

aufgefasst werden und ist einerseits in der Lage, einen unvollständigen Datentyp $y \in \mathcal{Y}$ zu analysieren, d.h. in seine Analysen $x \in X(y) \subseteq \mathcal{X}$ zu zerlegen; andererseits kann es für zwei alternative Analysen $x \neq x'$ vermittels einem Wahrscheinlichkeitsvergleich

$$p_\Theta(x) > p_\Theta(x')\ ,$$

eine Entscheidung für die Analyse x herbeiführen. Der Index Θ deutet an, dass diese Entscheidungsfindung von den Werten gewisser Parameter

$$\Theta \in \Omega$$

abhängt. Die Belegung dieser Parameterwerte findet im EM-Algorithmus durch eine iterative Anwendung von wohldefinierten *E- und M-Schritten* statt, die in jeder Iteration für einen gegenwärtigen Parameter $\Phi \in \Omega$ die *EM-Hilfsfunktion*

$$Q_\Phi(\Theta) = \sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in X(y)} p_\Phi(x|y) \cdot \log\ p_\Theta(x)$$

als eine *objektive Funktion* von Parametern $\Theta \in \Omega$ berechnen (E-Schritt) und einen *re-estimierten* Parameter

$$\hat{\Theta} = \text{argmax}_{\Theta \in \Omega} \; Q_{\Phi}(\Theta)$$

liefern (M-Schritt). Hierbei sind die *bedingten Wahrscheinlichkeiten*

$$p_{\Phi}(x|y) = \frac{p_{\Phi}(x)}{p_{\Phi}(y)}$$

und die *Wahrscheinlichkeiten der unvollständigen Datentypen*

$$p_{\Phi}(y) = \sum_{x \in X(y)} p_{\Phi}(x)$$

abgeleitete Wahrscheinlichkeiten der Modell-Wahrscheinlichkeiten $p_{\Theta}(x)$. (Siehe Abbildung 3.9: ein Pfeil markiert, dass die Quelle in der Definition des Ziels benötigt wird.) Es ist mehr als interessant, dass diese iterative Prozedur mit *zufällig gewählten Startparametern* gestartet wird und in der Praxis keine geschickte Wahl von Startparametern bekannt ist, mit der sich bessere Resultate als mit zufälligen Startparametern erzielen lassen (ausser für den Spezialfall der *Log-linearen Modelle* (Riezler et al. 2000)).

Zusammenfassend kann festgestellt werden, dass im stochastischen Ansatz des EM-Algorithmus ein symbolisches Analysewerkzeug dadurch verbessert werden soll, dass eine durch Erfahrung und Intuition vorgenommene Belegung bestimmter heuristischer Parameterwerte vermieden wird und stattdessen eine mathematisch wohldefinierte iterative Methode zur Auffindung optimaler Parameterwerte benutzt wird.

Interpretation des EM-Algorithmus

In diesem Abschnitt sollen die stochastischen Komponenten des EM-Algorithmus, insbesondere aber die EM-Hilfsfunktion, interpretiert werden. Als Schlüsselschritt wird sich hierbei die Einführung der sogenannten *empirischen Wahrscheinlichkeitsverteilung der vollständigen Datentypen* herausstellen.

Definition: Sei $\Phi \in \Omega$ ein vorgegebener Parameter des EM-Algorithmus. Dann heisst:

$$\tilde{p}_{\Phi} : \; \mathcal{X} \rightarrow [0,1] \qquad x \mapsto \tilde{p}_{\Phi}(x) \; ,$$

mit

$$\tilde{p}_{\Phi}(x) \; = \; \tilde{p}(y) \cdot p_{\Phi}(x|y) \qquad (\text{wobei } y \in \mathcal{Y} \text{ sodass } x \in X(y))$$

Korpus (oder *empirische Wahrscheinlichkeitsverteilung*) *der vollständigen Datentypen (unter dem gegenwärtigen Parameter Φ).*

y	y_1			y_2		y_3			
$\tilde{p}(y)$	0.3			0.2		0.5			
$X(y)$	x_{11}	x_{12}	x_{13}	x_{21}	x_{22}	x_{31}	x_{32}	x_{33}	x_{34}
$p_\Phi(x\|y)$	0.2	0.4	0.4	0.7	0.3	0.2	0.3	0.1	0.4
$\tilde{p}_\Phi(x)$	0.06	0.12	0.12	0.14	0.06	0.10	0.15	0.05	0.20

Abbildung 3.10: Beispiel einer empirischen Wahrscheinlichkeitsverteilung auf vollständigen Datentypen

Aus $0 \leq \tilde{p}(y),\ p_\Phi(x|y) \leq 1$ folgt offensichtlich $0 \leq \tilde{p}_\Phi(x) \leq 1$. Ferner gibt es, wegen der Bedingung

$$\mathcal{X} = \sum_{y \in \mathcal{Y}} X(y) \ ,$$

zu jedem vollständigen Datentyp $x \in \mathcal{X}$ genau ein $y \in \mathcal{Y}$ mit $x \in X(y)$. Die empirische Wahrscheinlichkeitsverteilung der vollständigen Datentypen ist daher wohldefiniert.

Beispiel: Da die Definition des *Korpus der vollständigen Datentypen (unter dem gegenwärtigen Parameter Φ)* nicht nur für den EM-Algorithmus, sondern auch für den SB-Algorithmus wichtig ist, wird in Abbildung 3.10 ein Beispiel angegeben. In der zweiten Zeile der Tabelle wird eine empirische Wahrscheinlichkeitsverteilung $\tilde{p}$ von unvollständigen Datentypen

$$\{\ y_1,\ y_2,\ y_3\ \} \subseteq \mathcal{Y}$$

angegeben, die ihrerseits in der ersten Zeile der Tabelle gezeigt werden. Diese Zahlen bedeuten, dass ein Korpus von unvollständigen Datentypen gegeben ist, und z.B. der unvollständige Datentyp y_1 mit 30% aller Vorkommen in ihm vertreten ist. In der dritten Zeile wird für jeden unvollständigen Datentyp $y \in \mathcal{Y}$ die Menge $X(y)$ aller seiner symbolischen Analysen angegeben. Z.B. hat der unvollständige Datentyp y_1 die drei vollständigen Analysen $x_{11},\ x_{12},\ x_{13}$. In der vierten Zeile wird eine (hier fiktive[5]) bedingte Wahrscheinlichkeitsverteilung $p_\Phi(x|y)$ von vollständigen Datentypen $x \in X(y)$, gegeben ihrem unvollständigen Datentyp $y \in \mathcal{Y}$, gezeigt. Auch diese Zahlen können als Vorkommenshäufigkeiten in einem Korpus gedeutet werden, wobei man allerdings davon ausgehen muss, dass in diesem Korpus nur ein einziger unvollständiger Datentyp y mit allen seinen Analysen $X(y)$ beobachtet wurde. Der Korpus des unvollständigen Datentyps y_1 und seiner Analysen $X(y_1)$ wird dann zu 40% aus der Analyse x_{13} bestehen. Aus diesen Vorga-

[5]Im EM-Algorithmus wird der Parameter Φ nur zu Beginn zufällig gesetzt, später mit einem mathematisch wohldefiniertem Kriterium estimiert. Die bedingte Wahrscheinlichkeitsverteilung $p_\Phi(x|y)$ ist daher nicht *fiktiv*, sondern wird mit dem Parameter Φ aus dem Modell (der parametrisierten Wahrscheinlichkeitsverteilungen) *berechnet*.

ben ergibt sich per Definition der Korpus $\tilde{p}_\Phi$. Er besteht höchstens aus den vollständigen Datentypen

$$\{\ x_{11},\ x_{12},\ x_{13},\ x_{21},\ x_{22},\ x_{31},\ x_{32},\ x_{33},\ x_{34}\ \} \subseteq \mathcal{X}$$

und wird in der fünften Zeile angegeben. In dem Korpus $\tilde{p}_\Phi$ kommt etwa der unvollständige Datentyp x_{13} mit 12% (= 30% · 40%) aller Vorkommen vor, während der unvollständige Datentyp x_{34} mit 20% aller Vorkommen am häufigsten, und der unvollständige Datentyp x_{33} mit 5% aller Vorkommen am seltensten ist.

Das Beispiel zeigt weiter, dass $\tilde{p}_\Phi$ nur auf der Menge

$$\sum_{y \in \mathcal{Y},\ \tilde{p}(y)>0} X(y)$$

Werte ungleich Null annehmen wird. Sind die Analysen $X(y)$ der unvollständigen Datentypen sämtlich endlich, so folgt, dass der Korpus der vollständigen Datentypen wie der Korpus der unvollständigen Datentypen ein endlicher Korpus sein wird.

Die *empirische Wahrscheinlichkeitsverteilung der vollständigen Datentypen* wird in Abbildung 3.9 in Relation zu anderen stochastischen Komponenten des EM-Algorithmus gezeigt. Die Abbildung ist so organisiert, dass die Wahrscheinlichkeitsverteilungen (Korpora) und stochastischen objektiven Funktionen getrennt dargestellt werden, je nachdem, ob sie unvollständige Datentypen (linker Teil der Abbildung) oder vollständige Datentypen (rechter Teil) beschreiben. Die Abbildung ist völlig symmetrisch aufgebaut: man entnimmt ihr zum Beispiel, dass $p_\Theta(y)$ für die unvollständigen Datentypen dieselbe Rolle spielt, wie $p_\Theta(x)$ für die vollständigen Datentypen. Ferner wird sich erweisen, dass $\tilde{p}_\Phi$ für die vollständigen Datentypen dieselbe Rolle spielt wie $\tilde{p}$ für die unvollständigen Datentypen. Insofern ist es gerechtfertigt, $\tilde{p}_\Phi$ als *Korpus* oder *empirische Wahrscheinlichkeitsverteilung der vollständigen Datentypen* zu bezeichnen, obwohl doch ganz offensichtlich die vollständigen Datentypen im Gegensatz zu den unvollständigen Datentypen in keinem realen, physischen Korpus vorliegen werden (was in vielen Fällen bewirkt, dass der EM-Algorithmus gegenüber überwachten Trainingsverfahren zu bevorzugen ist). Das folgende Lemma klärt das formale Verhältnis der beiden empirischen Wahrscheinlichkeitsverteilungen zueinander:

Lemma: Sei $\tilde{p}_\Phi$ die empirische Wahrscheinlichkeitsverteilung der vollständigen Datentypen. Dann gilt unabhängig vom gegenwärtigen Parameter $\Phi \in \Omega$:

$$\tilde{p}(y) \ = \sum_{x \in X(y)} \tilde{p}_\Phi(x) \qquad (y \in \mathcal{Y}).$$

Beweis:

$$\sum_{x \in X(y)} \tilde{p}_\Phi(x) \;=\; \sum_{x \in X(y)} \tilde{p}(y) \cdot p_\Phi(x|y) \;=\; \tilde{p}(y) \sum_{x \in X(y)} p_\Phi(x|y) \;=\; \tilde{p}(y) \;.$$

q.e.d.

Das Lemma zeigt, dass die empirische Wahrscheinlichkeitsverteilung der vollständigen Datentypen nicht nur dem Namen nach eine Wahrscheinlichkeitsverteilung ist:

$$\sum_{x \in \mathcal{X}} \tilde{p}_\Phi(x) \;=\; \sum_{y \in \mathcal{Y}} \sum_{x \in X(y)} \tilde{p}_\Phi(x) \;=\; \sum_{y \in \mathcal{Y}} \tilde{p}(y) \;=\; 1 \;.$$

Das nächste Lemma ist von zentraler Bedeutung für die Interpretation des EM-Algorithmus. Mit Hilfe der empirischen Wahrscheinlichkeitsverteilung $\tilde{p}_\Phi$ über den vollständigen Datentypen gelingt es, die EM-Hilfsfunktion als eine *Log-Likelihood auf dem Korpus* $\tilde{p}_\Phi$ zu deuten, sowie den EM-Algorithmus als eine iterative Maximum-Likelihood-Estimierung von *Log-Likelihood-Funktionen auf vollständigen Datentypen* zu identifizieren.

Lemma: Sei $\tilde{p}_\Phi$ die empirische Wahrscheinlichkeitsverteilung der vollständigen Datentypen und L_Θ die Log-Likelihood der vollständigen Datentypen auf dem Korpus $\tilde{p}_\Phi$, d.h.

$$L_\Phi(\Theta) \;=\; \sum_{x \in \mathcal{X}} \tilde{p}_\Phi(x) \cdot \log\; p_\Theta(x) \;.$$

Dann gilt für alle $\Theta \in \Omega$:

$$Q_\Phi(\Theta) \;=\; L_\Phi(\Theta) \;.$$

Beweis:

$$\begin{aligned}
L_\Phi(\Theta) &= \sum_{x \in \mathcal{X}} \tilde{p}_\Phi(x) \cdot \log\; p_\Theta(x) \\
&= \sum_{y \in \mathcal{Y}} \sum_{x \in X(y)} \tilde{p}_\Phi(x) \cdot \log\; p_\Theta(x) \\
&= \sum_{y \in \mathcal{Y}} \sum_{x \in X(y)} (\tilde{p}(y) \cdot p_\Phi(x|y)) \cdot \log\; p_\Theta(x) \\
&= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in X(y)} p_\Phi(x|y) \cdot \log\; p_\Theta(x) \\
&= Q_\Phi(\Theta)
\end{aligned}$$

q.e.d.

Abbildung 3.9 zeigt den im Lemma vorgestellten Zusammenhang in grösserer Klarheit:

- Für die unvollständigen Datentypen (linker Teil der Abbildung) gilt: Der Korpus $\tilde{p}(y)$ und die (Logarithmen der) Wahrscheinlichkeitsverteilung $p_\Theta(y)$ ergeben die

Eingabe: wie in Abbildung 3.7

Ausgabe: dito

Prozedur: * Modifizierte 5. und 6. Zeile des EM-Algorithmus *\

5a. **generate** corpus $\tilde{p}_\Phi(x)$ of complete data (given parameter Φ)

5b. **compute** log-likelihood $L_\Phi(\Theta)$ of model $p_\Theta(x)$ on corpus $\tilde{p}_\Phi(x)$

6. **compute** $\hat{\Theta} = \text{argmax}_{\Theta \in \Omega} \; L_\Phi(\Theta)$

Abbildung 3.11: Der EM-Algorithmus unter Benutzung des Korpus und der Log-Likelihood der vollständigen Datentypen

Eingabe: wie in Abbildung 3.7

Ausgabe: dito

Prozedur: * Modifizierte 5. und 6. Zeile des EM-Algorithmus *\

5. **generate** corpus $\tilde{p}_\Phi(x)$ of complete data (given parameter Φ)

6. **compute** maximum-likelihood estimate $\hat{\Theta}$ of model $p_\Theta(x)$ on corpus $\tilde{p}_\Phi(x)$

Abbildung 3.12: Der EM-Algorithmus als iterative Maximum-Likelihood-Estimierung auf dem Korpus der vollständigen Datentypen

Log-Likelihood $L(\Theta)$ des EM-Algorithmus. Der EM-Algorithmus wurde *konzipiert*, um diese Log-Likelihood als Funktion der Parameter Θ zu maximieren (Dempster et al. 1977).

- Für die vollständigen Datentypen (rechter Teil der Abbildung) gilt analog: Der Korpus $\tilde{p}_\Phi(x)$ und die (Logarithmen der) Wahrscheinlichkeitsverteilung $p_\Theta(x)$ ergeben die Log-Likelihood $L_\Phi(\Theta)$. Der EM-Algorithmus wurde *realisiert*, indem er bei der Abarbeitung eines kombinierten E- und M-Schritts (bzw. der Ausführung der EM--Parameter-Abbildung $\mathcal{M}(.)$) die EM-Hilfsfunktion $Q_\Phi(\Theta)$ als Funktion der Parameter Θ maximiert. Da die EM-Hilfsfunktion und die Log-Likelihood $L_\Phi(\Theta)$ gemäss Lemma übereinstimmen, wird in jedem Schritt des EM-Algorithmus also eine Maximum-Likelihood-Estimierung auf vollständigen Datentypen ausgeführt (weitere Details siehe unten).

Dieses Lemma hat drei Auswirkungen, von denen die ersten zwei in diesem Abschnitt und

die letzte und folgenreichste im nächsten Abschnitt besprochen werden.

(i) Abbildung 3.11 zeigt eine leicht veränderte Implementierung des EM-Algorithmus (vergleiche Abbildung 3.7), wobei allerdings nur die modifizierten Zeilen gezeigt werden. Da die EM-Hilfsfunktion mit der Log-Likelihood des Modells $p_\Theta(x)$ auf dem Korpus $\tilde{p}_\Phi(x)$ der vollständigen Datentypen übereinstimmt, darf die 5. Zeile der Prozedur des EM-Algorithmus in zwei seperate Arbeitsgänge aufgeteilt werden: zunächst wird der Korpus $\tilde{p}_\Phi(x)$ generiert (Zeile 5a), anschliessend wird auf ihm die Log-Likelihood $L_\Phi(\Theta)$ des Modells $p_\Theta(x)$ berechnet (Zeile 5b). Natürlich muss nach diesen Veränderungen auch die 6. Zeile des EM-Algorithmus modifiziert werden, sodass die Log-Likelihood $L_\Phi(\Theta)$ anstelle der EM-Hilfsfunktion $Q_\Phi(\Theta)$ maximiert wird.

(ii) Abbildung 3.12 zeigt eine weitere leichte, allerdings entscheidende Modifikation der Implementierung des EM-Algorithmus. Wieder werden nur die veränderten Zeilen gezeigt. Während die Zeile 5a des modifizierten Algorithmus in Abbildung 3.11 unverändert als neue Zeile 5 übernommen wird, ist die Schlüssel-Beobachtung, dass die Zeilen 5b und 6 eine Maximum-Likelihood-Estimierung darstellen. Diese zwei Zeilen können also zu der neuen Zeile 6 zusammengefasst werden.

Als sehr schönes Resultat gewinnt man die Erkenntnis, dass sich der EM-Algorithmus als eine iterative Anwendung von Maximum-Likelihood-Estimierungen darstellen lässt, wobei es sehr angenehm erscheint, dass jede dieser Maximum-Likelihood-Estimierungen auf einem Korpus $\tilde{p}_\Phi$ von vollständigen Datentypen ausgeführt wird.

3.4 Der EM-Algorithmus als Bootstrap-Algorithmus

Das Ziel dieses Abschnitts ist es, noch einen Schritt über die zwei Ziele des vorangegangenen Abschnitts hinauszugehen und den EM-Algorithmus als eine besonders gut motivierte Instanz eines sehr viel allgemeineren Algorithmus vorzustellen. Dieser Algorithmus könnte vielleicht als *stochastischer Bootstrap-Algorithmus (SB-Algorithmus)* bezeichnet werden und ist insofern interessant, da er für beliebige stochastische Modelle ein iteratives *nicht-überwachtes Trainingsverfahren* (auf unvollständigen Datentypen) darstellt, wobei lediglich vorausgesetzt werden muss, dass die stochastischen Modelle mit einem *überwachten Trainingsverfahren* (auf vollständigen Datentypen) prinzipiell trainierbar sind. (Die Wortwahl *prinzipiell trainierbar* soll ausdrücken, dass keine vollständige Trainingsdaten für das überwachte Trainingsverfahren zur Verfügung stehen müssen, sondern nur bekannt sein muss, wie trainiert werden würde, wenn eine ausreichende Menge vollständiger Trainingsdaten vorhanden wäre.) Als Schlüsselschritt des SB-Algorithmus fungiert erneut die *empirische Wahrscheinlichkeitsverteilung der vollständigen Datentypen.* Der SB-Algorithmus kombiniert diese Wahrscheinlichkeitsverteilung mit einem vorgegebenen überwachten Trainingsverfahren, um es in ein iteratives unüberwachtes Trainingsverfahren zu überführen. Der Vorteil des SB-Algorithmus ist, dass es mit ihm unproblematisch ist, viele Modelle auf riesigen Trainingskorpora zu trainieren. Der Nachteil des SB-Algorithmus liegt darin, dass er eine (zumindest einfache) *symbolische Analysekomponente* voraussetzt. Ferner besitzt der SB-Algorithmus möglicherweise nicht alle guten (vielleicht aber auch nicht alle schlechten) Eigenschaften des zugrundeliegenden überwachten Trainingsverfahrens.

Im vorherigen Abschnitt wurde gezeigt, dass sich der EM-Algorithmus als eine iterative Anwendung von Maximum-Likelihood-Estimierungen auf vollständigen Datentypen darstellen lässt. Natürlich ist die Maximum-Likelihood-Estimierung nur ein Verfahren unter vielen möglichen Trainingsverfahren, um zu einer Parameter-Estimierung zu gelangen (siehe Abschnitt 2.4). Somit ist es legitim, darüber nachzudenken, diejenigen Teile des EM-Algorithmus aus Abbildung 3.12, die eine Maximum-Likelihood-Estimierung für vollständige Datentypen ausführen, durch beliebige alternative Parameter-Estimierungsverfahren zu ersetzen, um möglicherweise völlig neue iterative Trainingsverfahren für unvollständige Datentypen zu erhalten. Denn offensichtlich führt dieser Typ von Modifikationen nicht mehr nur zu unterschiedlichen Implementierungen derselben Funktionalität (wie im vorherigen Abschnitt), sondern verändert in nicht leicht vorhersehbarer Weise die Eigenschaften des Algorithmus. Die angesprochene unkontrollierte Modifikationsfreiheit ist aber zumindest in einer Hinsicht sehr nützlich: Sie ermöglicht es prinzipiell jedem überwachten stochastischen Lernverfahren (auf vollständigen Datentypen) zu einem iterativen, aber unüberwachten stochastischen Lernverfahren (auf unvollständigen Datentypen) erweitert

Eingabe: Ein *Trainingskorpus* $\tilde{p}(y)$ von unvollständigen Datentypen $y \in \mathcal{Y}$, *Modelle* $p_\Theta(x)$ für die vollständigen Datentypen $x \in \mathcal{X}$, eine endliche Menge Ω_0 von *Startparametern*, eine *Iterationsanzahl* $numberOfIterations \geq 1$, eine *symbolische Analyse-Komponente*

$$X : \mathcal{Y} \to \mathcal{P}(\mathcal{X})$$

mit $|X(y)| < \infty$, $(y \in \mathcal{Y})$, sowie ein beliebiges *überwachtes Trainingsverfahren*

$$\delta : \mathcal{F}(\mathcal{X}) \to \Omega$$

für die Modelle $p_\Theta(x)$. D.h. $\mathcal{F}(\mathcal{X})$ ist die Menge der endlichen Korpora, die aus vollständigen Datentypen $x \in \mathcal{X}$ bestehen.

Ausgabe: Für jeden Startwert $\Theta_0 \in \Omega_0$ eine Folge von re-estimierten Parametern:

$$\Theta_1,\ \Theta_2,\ \Theta_3,\ \ldots \in \Omega\ .$$

Prozedur:

1. * Version mit Variation der Startparameter: *\
2. **for each** $\Theta_0 \in \Omega$ **do**
3. **for each** $i := 1, \ldots,$ *numberOfIterations* **do**
4. $\Phi := \Theta_{i-1}$;
5. **generate** corpus $\tilde{p}_\Phi(x)$ of complete data $x \in \mathcal{X}$;
6. **compute** $\hat{\Theta} = \delta\left(\tilde{p}_\Phi\right)$ exploiting supervised training on $\tilde{p}_\Phi$;
7. $\Theta_i := \hat{\Theta}$;
8. **print** $\Theta_0,\ \Theta_1,\ \Theta_2,\ \Theta_3,\ \ldots$;

Abbildung 3.13: SB-Algorithmus

zu werden. Der diese Erweiterung beschreibende Algorithmus könnte vielleicht recht treffend als *stochastischer Bootstrap-Algorithmus (SB-Algorithmus)* bezeichnet werden.

Abbildung 3.13 zeigt den SB-Algorithmus aus prozeduraler Sicht. Seine Eingabe besteht wie die Eingabe des EM-Algorithmus aus einem Trainingskorpus $\tilde{p}(.)$ von unvollständigen Datentypen, aus parametrisierten vollständigen Wahrscheinlichkeitsverteilungen $p_\Theta(.)$, aus einer Menge $\Omega_0 \subseteq \Omega$ von Startparametern, aus einer (im allgemeinen relativ hohen) Anzahl $numberOfIterations \geq 1$ von festgelegten Iterationen, sowie einer symbolischen Analyse-Komponente $X(.)$, die zu jedem unvollständigen Datentyp $y \in \mathcal{Y}$ die Menge $X(y)$ seiner Analysen bereitstellt, wobei vorausgesetzt wird, dass kein unvollständiger Datentyp eine unendliche Anzahl von Analysen $X(y)$ besitzt. Zusätzlich zur Eingabe des EM-Algorithmus wird verlangt, dass ein sogenanntes *überwachtes Trainingsverfahren* δ für vollständige Datentypen vorliegt.

Die Ausgabe des SB-Algorithmus besteht (für jeden *Startparameter* $\Theta_0 \in \Omega_0$) aus einer Folge von re-estimiertemn Parametern

$$\Theta_1,\ \Theta_2,\ \Theta_3,\ \ldots \in \Omega\ ,$$

d.h. aus Parametern, die durch eine iterierte Anwendung des überwachtes Trainingsverfahrens δ auf den Korpus der vollständigen Datentypen entstehen:

$$\Theta_i = \delta\left(\tilde{p}_{\Theta_{i-1}}\right) \qquad (i = 1, 2, 3, \ldots)\ .$$

Leider können (ohne zusätzliche Voraussetzungen an die Estimierungsmethode δ) für den SB-Algorithmus keinerlei Versprechen hinsichtlich der Konvergenz der Log-Likelihood der unvollständigen Datentypen gegen globale oder lokale Maxima gemacht werden.

In der 2. Zeile der Prozedur des SB-Algorithmus wird eine Schleife über die Startparameter initialisiert, deren Schritte in der 3. bis 8. Zeile ausgeführt werden. In der 3. Zeile wird eine Schleife über die Trainingsiterationen initialisiert, deren Schritte in der 4. bis 7. Zeile ausgeführt werden. In der 4. Zeile wird der *gegenwärtige Parameter* Φ für die Generierung des Korpus $\tilde{p}_\Phi$ der vollständigen Datentypen gesetzt. Die Korpus-Generierung findet in der 5. Zeile statt. In der 6. Zeile wird mit dem gegebenen überwachten Trainingsverfahren δ und dem fertig generierten Korpus $\tilde{p}_\Phi$ der *neue Parameter* $\hat{\Theta}$ bestimmt. In der 7. Zeile wird der neue Parameter gespeichert, um ihn für die nächste Iteration (als gegenwärtigen Parameter) benutzen zu können. In der 8. Zeile wird für jeden Startparameter die Folge der aus ihm berechneten Parameter ausgegeben.

Zusammenfassend kann man feststellen, dass die *empirische Wahrscheinlichkeitsverteilung der vollständigen Datentypen* als Schlüsselschritt des SB-Algorithmus fungiert. Mit dem SB-Algorithmus ist es völlig unproblematisch, Modelle auf riesigen Trainingskorpora

zu trainieren, da diese lediglich aus unvollständigen Datentypen bestehen müssen. Der SB-Algorithmus besitzt diesen Vorteil, weil er annimmt, dass eine *symbolische Analysekomponente* existiert, sodass die Analysen des gegebenen Trainingskorpus (mit Hilfe des gegenwärtig "besten" Parameters, bzw. seines Modells) dazu benutzt werden können, einen "angemessenen" Korpus von vollständigen Datentypen zu generieren. Der SB-Algorithmus nutzt die empirische Wahrscheinlichkeitsverteilung des generierten Korpus, um daraus mit einem *vorgegebenen überwachten Trainingsverfahren auf vollständigen Datentypen* einen (hoffentlich verbesserten) neuen Parameter zu bestimmen.

Eine iterative Anwendung dieses Vorgangs bringt eine Folge von estimierten Parametern hervor, aus denen mit Hilfe einer linguistischen Evaluierung ein optimaler Parameter ausgewählt werden kann. Ohne zusätzliche Voraussetzungen ist es leider unklar, ob der SB-Algorithmus die guten (vielleicht aber auch die schlechten) Eigenschaften des zugrundeliegenden überwachten Trainingsverfahrens erbt.

Dieser Abschnitt soll mit der erhellenden Bemerkung geschlossen werden, dass der EM-Algorithmus eine besonders gut motivierbare Instanz des SB-Algorithmus ist, die man dadurch erhält, dass im SB-Algorithmus als zugrundeliegendes überwachtes Trainingsverfahren die Maximum-Likelihood-Estimierung auf vollständigen Datentypen benutzt wird.

3.5 Training mit dem EM-Algorithmus

Abbildung 3.14 zeigt das Training von Modellen $p_\Theta(.)$ mit dem EM-Algorithmus, sowie die systematische Evaluierung aller seiner freien Parameter: Trainingskorpusgrösse, Anzahl Trainingsiterationen, Startparameter. In der Abbildung sind die Algorithmen mit dem Symbol ▭, die Daten mit dem Symbol ⛁ gekennzeichnet.

Die Abbildung 3.14 teilt sich in einen linken Teil (gestrichelt umrahmt) und einen rechten Teil.

Im rechten Teil der Abbildung wird die Aufteilung eines gegebenen, oft mehrere 100 Millionen Datentoken umfassenden Korpus von unvollständigen Datentypen gezeigt, die in einen Trainingskorpus (unvollständige Datentypen), einen Entwicklungs-Testkorpus (vollständige Datentypen), sowie einen Testkorpus (ebenfalls vollständige Datentypen) resultiert. Die *Aufteilung* des Korpus in einen *Entwicklungskorpus* und einen *Testkorpus* bereitet den Test vor, wie gut das trainierte Modell arbeitet. Um die Fähigkeit zur Generalisierung zu testen, sollten die Testdaten ungesehene Daten sein, d.h. Daten die nicht für die Entwicklung verwendet wurden. In der Regel wird die Aufteilung des gegebenen Korpus in Entwicklungs- und Testdaten so vorgenommen, dass man 90% Entwicklungsdaten und 10% Testdaten erhält. Natürlich sollten die Testdaten unabhängig von den Entwicklungsdaten sein, aber diesen nicht zu unähnlich sein. Die übliche Prozedur, um dies zu erreichen, ist es, jedes zehnte Datentoken des gegebenen Korpus als Testdatum zu nehmen. Da während der Entwicklungsphase mehrere Modellversionen anfallen, aber nur eine einziges Modell "entwickelt" werden soll, ist es nötig, dass der Entwicklungskorpus seinerseits aufgeteilt wird. Das Ergebnis dieser Aufteilung, die wie oben vorgenommen werden kann, indem jeder zehnte Datentoken des Entwicklungskorpus als Entwicklungs-Testdatum benutzt wird, sind 90% Trainingskorpus und 10% Entwicklungs-Testkorpus. Da es sich bei dem Testkorpus und dem Entwicklungs-Testkorpus um unvollständige Datentypen handelt, müssen beide Testkorpora annotiert werden. Leider ist die manuelle Herstellung von annotierten Testkorpora sehr aufwendig. Es ist üblich, jeweils mindestens 300 - 600 unvollständigDatentypen zu annotieren[6]. Je grösser diese Zahl ist, desto kleinere Auswirkungen werden gelegentliche Annotationsfehler haben, und desto robuster werden die Evaluierungs-Ergebnisse sein. Da die Annotation der Testdaten erhebliche Kosten verursacht, ist es sinnvoll, wenn auch nicht ungefährlich, die Anzahl der Testdaten möglichst klein zu halten. Das Motto für die Erstellung dieser Daten lautet also: so gross und robust, wie nötig, aber so klein und kostengünstig, wie möglich. Zur Entscheidungsfindung, wieviele Daten nötig sind, um robuste Evaluierungsergebnisse zu erhalten, kann

[6]Diese Zahl ist sehr aufgabenspezifisch. Zum Beispiel wären 600 Wörter für einen Tagger etwas wenig.

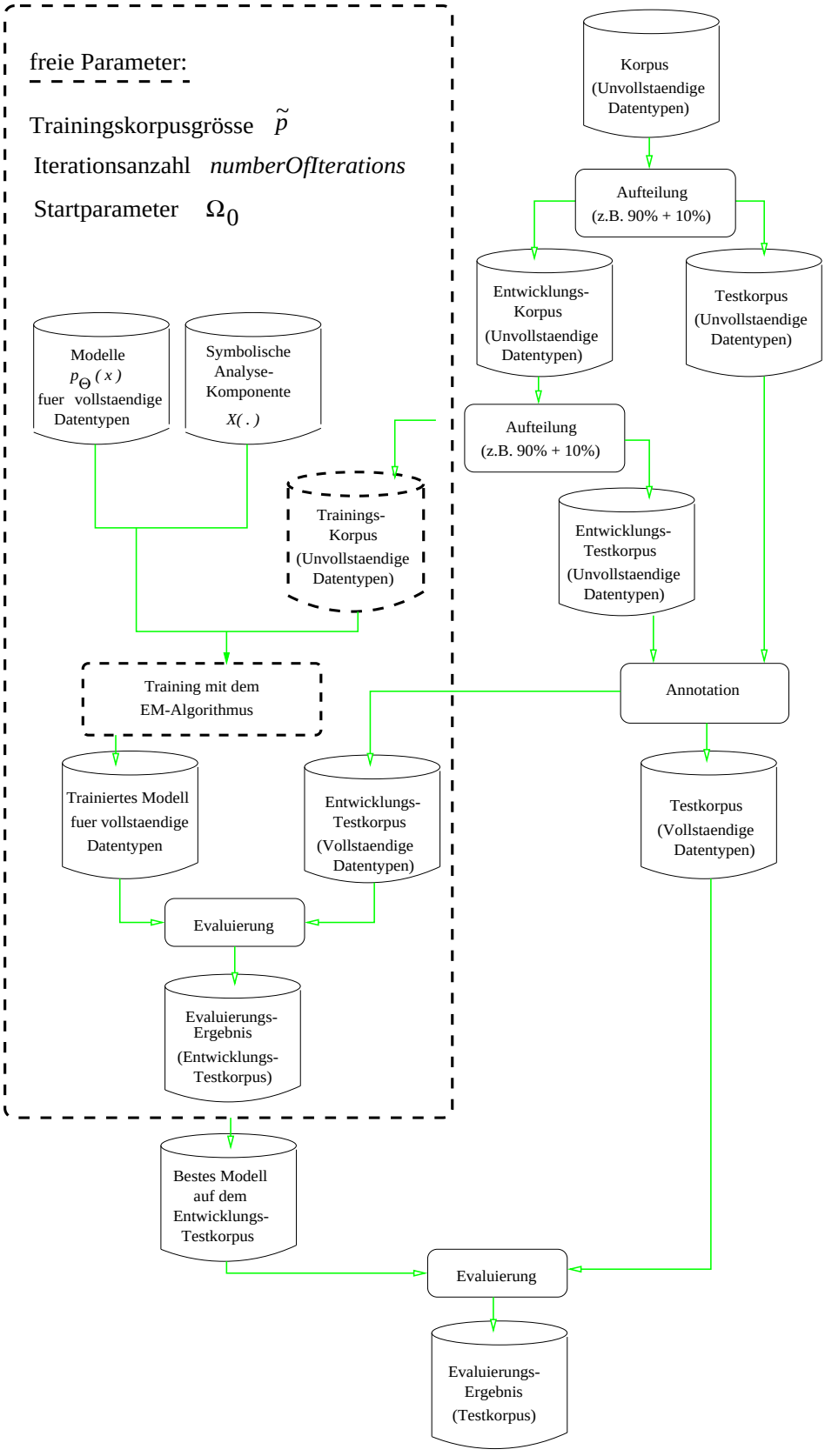


Abbildung 3.14: Training und Evaluierung mit dem EM-Algorithmus

die hervorragende Arbeit von Yeh (2000) konsultiert werden.

Im linken Teil der Abbildung 3.14 wird der EM-Algorithmus aus Abbildung 3.7 innerhalb einer *Schleife über alle Parameter* angewendet. Die freien Parameter sind in der Abbildung gestrichelt dargestellt: es handelt sich um: (i) die *Startparameter*, (ii) den *Trainingskorpus (unvollständige Datentypen)*, sowie (iii) die *Anzahl der im EM-Algorithmus vorzunehmenden Iterationen*. Naürlich sind Zahlenangaben zu der Anzahl der freien Parameter unseriös, da diese stets von der Komplexität der gegebenen Modelle (Parser, Tagger, Sprachmodell, u.s.w.), aber auch von praktischen Gesichtspunkten, wie einem eventuell für die Evaluierung vorgegebenen Zeitrahmen abhängen: Die Zahl der Startparameter kann eigentlich nie hoch genug sein und sollte nur durch die Rechenzeit begrenzt werden, die man zur Verfügung hat, um ein Ergebnis vorzuweisen: 10 bis 100 Startparameter sollten ausprobiert werden. Die Zahl der Iterationen ist oft unproblematisch, da gelegentlich schon nach wenigen Iterationen das beste Modell generiert wird. Um dies nachzuweisen wird man aber doch im allgemeinen etwa 5 bis 50 Iterationen vornehmen. Äusserst mangelhaft ist es, mit einer festen Anzahl von Iterationen zu arbeiten, da die Güte des Modells in der Regel schnell erreicht wird, aber genauso schnell wieder sinkt. Arbeitet man also mit einer festen Iterationsanzahl, so arbeitet man fast ausschliesslich mit schlechten Modellen. Die Grösse des Trainingskorpus zu variieren ist für den EM-Algorithmus weniger sinnvoll als für überwachte Trainingsverfahren, die aus Kostengründen oft darauf angewiesen sind mit möglichst kleinen Trainingskorpora auszukommen. Im allgemeinen wird trotzdem mit verschiedenen Trainingskorpora experimentiert werden, etwa von der Grösse $\frac{1}{10}\times$, $\frac{1}{100}\times$, $\frac{1}{1000}\times, \ldots$ des Ausgangskorpus. Für jede Kombination aller freien Parameter wird der EM-Algorithmus aus Abbildung 3.7 benutzt, um trainierte Modelle zu berechnen. Jedes der resultierenden Modelle wird mit Hilfe des Entwicklungs-Testkorpus evaluiert. Die Evaluierung eines Modells ist stets eine schwierige Aufgabe. Die Standard-Vorgehensweise ist, den Entwicklungs-Testkorpus unter Benutzung der symbolischen Analysekomponente zu parsen und die vom gegebenen Modell ermittelte wahrscheinlichste Analyse (eines jeden unvollständigen Datentyps) mit der annotierten Analyse zu vergleichen. Hierfür stehen je nach Anwendung eine Reihe von Evaluierungsmetriken zur Verfügung. Als Ausgabe der Schleife über die freien Parameter erhält man, nach der Evaluierung aller trainierten Modelle, das auf dem Entwicklungs-Testkorpus beste Modell. Dieses Modell könnte man das *mit dem EM-Algorithmus trainierte Modell* nennen. Eine abschliessende Evaluierung dieses Modells auf dem Testkorpus liefert schliesslich das Evaluierungs-Ergebnis, welches reportiert werden darf.

3.6 Formale Eigenschaften des EM-Algorithmus

Die originale Absicht bei der Einführung des EM-Algorithmus war es, eine iterative Berechnung der Maximum-Likelihood-Estimierung vornehmen zu können (Dempster et al. 1977). Trotz seines langsamen Konvergenzverhaltens (siehe unten, Theorem 4 von Dempster et al. (1977)) wurde der EM-Algorithmus schon in den 70er Jahren zu einem sehr populären numerischen Verfahren innerhalb der Statistik. Im Gegensatz zu den damaligen Erfahrungen mit numerischen Optimierungsmethoden, war der EM-Algorithmus sehr einfach zu implementieren: Lösungen des M-Schritts existierten oft in geschlossener Form, oder konnten mit Standard-Programmpaketen aus der Statistik gelöst werden. Ein weiterer Vorteil des EM-Algorithmus war, dass er nur sehr wenig Speicherplatz benötigte. Während der letzte Aspekt in der Zwischenzeit an Bedeutung verloren hat, wird der EM-Algorithmus heute typischerweise für Probleme eingesetzt, von denen man den Eindruck hat, dass sie ohne Benutung des EM-Algorithmus nur schwer (bzw. garnicht) lösbar sind, oder wenn alternative Lösungsmethoden sehr hohe Kosten verursachen würden (beispielsweise weil sie manuell annotierte Daten benötigen). Aus diesem Grund ist der EM-Algorithmus nach wie vor in der Statistik sehr beliebt, während in der stochastischen Linguistik damit begonnen wird, ihn bewusst für aktuelle Probleme einzusetzen.

In diesem Abschnitt sollen die grundlegenden Eigenschaften des EM-Algorithmus vorgestellt werden, sowie die Frage beantwortet werden, ob der EM-Algorithmus tatsächlich seinen Job macht und eine iterative Maximum-Likelihood-Estimierung vornimmt. Hierzu werden die wichtigsten Theoreme in Dempster et al. (1977) und Wu (1983) vorgestellt werden. Diese Theoreme werden um einige neue Resultate ergänzt, die für sogenannte *injektive Parametrisierungen der Analysen* $X(y)$ in der Lage sind, die Korollare von Dempster et al. (1977) und Wu (1983) über Log-Likelihood-Funktionen mit eindeutigen Maxima zu ersetzen.

Einleitende Bemerkungen zur Maximum-Likelihood-Estimierung im Kontext des EM-Algorithmus

In Kapitel 2 wurde die Maximum-Likelihood-Estimierung als ein Optimierungsproblem, nämlich als das Problem der Maximierung der Korpuswahrscheinlichkeit (bzw. der parametrisierten Log-Likelihood) vorgestellt:

$$\begin{aligned} \hat{\Theta} &= \text{argmax}_{\Theta \in \Omega} \; L(\Theta) \\ &= \text{argmax}_{\Theta \in \Omega} \; \sum_{y \in \mathcal{Y}} \tilde{p}(y) \cdot \log \; p_\Theta(y) \; . \end{aligned}$$

Abbildung 3.15 skizziert das Problem. Auf der x-Achse werden die Parameter $\Theta \in \Omega$

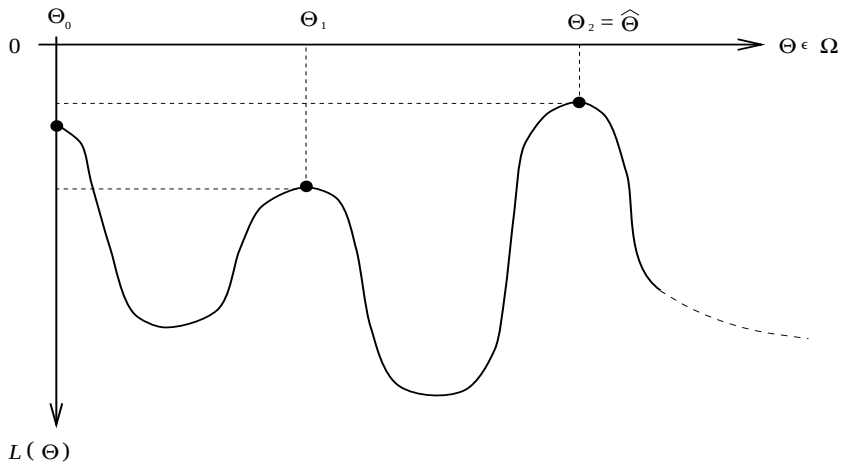


Abbildung 3.15: Maximum-Likelihood-Estimierung

abgetragen, auf der y-Achse die Werte der Log-Likelihood $L(\Theta)$. Da die Wahrscheinlichkeiten $p_\Theta(y)$ kleiner oder gleich 1 sind, sind die Terme $\log\ p_\Theta(y)$ kleiner oder gleich 0. Die Log-Likelihood nimmt deshalb nur negative Werte an. Die globale Maximierung einer Funktion mit mehrdimensionalen Argumenten, welche eventuell gewisse Nebenbedingungen erfüllen sollen, ist ein sehr schwieriges mathematisches Problem. Es wird in der Regel nach folgendem Muster vorgenommen:

1. Die Paramter aller *lokalen Maxima* der Log-Likelihood werden bestimmt. In Abbildung 3.15 sind dies Θ_1 und Θ_2.

2. Der sogenannte *Rand* $\partial\Omega$ des Parameterraums Ω wird bestimmt. In Abbildung 3.15 sind dies die Parameter Θ_0 und $+\infty$.

3. Der optimale Parameter $\hat{\Theta}$ der Maximum-Likelihood-Estimierung ist der Parameter, der zu dem grössten lokalen Maximum, bzw. zu dem grössten Randwert der Log-Likelihood gehört. In Abbildung 3.15 ist dies der Parameter Θ_2.

Wie der EM-Algorithmus könnte auch die Maximum-Likelihood-Estimierung als ein Erwartungswert-Maximierungs-Verfahren bezeichnet werden, da nach der Berechnung gewisser, relevanter Erwartungswerte:

$$L:\ \Omega \to \mathbb{R}, \quad \Theta \mapsto \tilde{p}\,[\ \log\ p_\Theta\ ]$$

eine entsprechende Maximierung erfolgt:

$$\begin{aligned} \hat{\Theta} &= \text{argmax}_{\Theta\in\Omega}\ \ L(\Theta) \\ &= \text{argmax}_{\Theta\in\Omega}\ \ \tilde{p}\,[\ \log\ p_\Theta\ ]\ . \end{aligned}$$

	Maximum-Likelihood-Estimierung (auf unvollständigen Datentypen $\mathcal{Y}$)	**EM-Algorithmus** (auf vollständigen Datentypen $\mathcal{X}$ vermittels der symbolischen Analyse-Komponente $X(.)$)
$L(\Theta)$	$\sum_{y\in\mathcal{Y}} \tilde{p}(y) \cdot \log\ p_\Theta(y)$	$\sum_{y\in\mathcal{Y}} \tilde{p}(y) \cdot \log \sum_{x\in X(y)} p_\Theta(x)$
$Q_\Phi(\Theta)$	-	$\sum_{y\in\mathcal{Y}} \tilde{p}(y) \sum_{x\in X(y)} p_\Phi(x\|y) \cdot \log\ p_\Theta(x)$
$\hat{\Theta}$	$\text{argmax}_{\Theta\in\Omega}\quad L(\Theta)$	$\text{argmax}_{\Theta\in\Omega}\quad Q_\Phi(\Theta)$

Abbildung 3.16: Der EM-Algorithmus als iteratives Lösungsverfahren der Maximum-Likelihood-Estimierung

In vielen Fällen ist eine Lösung dieses Optimierungsproblems bekannt. So wurde zum Beispiel in Abschnitt 2.4 gezeigt, dass die Parametrisierung $p_\Theta(y) = \Theta_y$ zu der naheliegenden Lösung $\hat{\Theta}_y = \tilde{p}(y)$ führt ($y \in \mathcal{Y}$). Eine einfache, aber sehr wichtige Schlüssel-Beobachtung ist, dass der EM-Algorithmus annimmt, dass sich die Wahrscheinlichkeiten der unvollständigen Datentypen $y \in \mathcal{Y}$ auf der Grundlage der Wahrscheinlichkeiten bestimmter vollständiger Datentypen $x \in X(y)$ erklären lassen. Für den EM-Algorithmus hat dies zur Folge, dass keine direkte Parametrisierung der unvollständigen Wahrscheinlichkeitsverteilungen $p(y) = \sum_{x\in X(y)} p(x)$ vorhanden ist, sondern sich aus der für die vollständigen Wahrscheinlichkeitsverteilungen $p(x)$ vorgenommenen Parametrisierung errechnet. Leider zeigt die nächste Formel, dass sich aus diesem Grund die Maximum-Likelihood-Estimierung für den EM-Algorithmus formal deutlich schwieriger gestaltet als im einleitenden Beispiel:

$$\begin{aligned}\hat{\Theta} &= \text{argmax}_{\Theta\in\Omega} \quad \sum_{y\in\mathcal{Y}} \tilde{p}(y) \cdot \log\ p_\Theta(y) \\ &= \text{argmax}_{\Theta\in\Omega} \quad \sum_{y\in\mathcal{Y}} \tilde{p}(y) \cdot \log \sum_{x\in X(y)} p_\Theta(x)\ .\end{aligned}$$

Obwohl die Maximierung des Logarithmus einer Summe nicht grundsätzlich unmöglich ist, stellt sie doch ein mathematisch anspruchsvolles Problem dar. Es ist interessant, dass der EM-Algorithmus als ein iteratives Lösungsverfahren für dieses Problem angesehen werden kann[7]. Abbildung 3.16 zeigt einen Vergleich der Maximum-Likelihood-Estimierung (zweite Spalte) mit dem EM-Algorithmus (dritte Spalte). Es werden jeweils die Log-Likelihood (zweite Zeile), die EM-Hilfsfunktion (dritte Zeile), und das Optimierungskriterium des

[7] mit der Konsequenz, dass der EM-Algorithmus nicht angewendet werden sollte, wenn man die Lösung der Maximum-Likelihood-Estimierung direkt finden kann.

estimierten Parameters (vierte Zeile) angegeben. Der Vergleich zeigt:

1. Der Term $\log \sum_{x \in X(y)} p_\Theta(x)$ (der Logarithmus einer Summe), der in der Log-Likelihood $L(\Theta)$ des EM-Algorithmus vorkommt und als Term

   $$\log\ p_\Theta(y)$$

   in der Log-Likelihood der Maximum-Likelihood-Estimierung, wird in der EM-Hilfsfunktion $Q_\Phi(\Theta)$ durch den Term $\sum_{x \in X(y)} p_\Phi(x|y) \cdot \log\ p_\Theta(x)$ (eine Summe von Logarithmen) ersetzt, der als bedingte Erwartung

   $$p_\Phi(.|y)\,[\ \log\ p_\Theta\ ]$$

   interpretierbar ist. Der gegenwärtige Parameter Φ sollte hierbei als eine Annäherung an den Parameter $\hat{\Theta}$ der Maximum-Likelihood-Estimierung angesehen werden.

2. In der Maximum-Likelihood-Estimierung ist die Maximierung der Log-Likelihood (d.h. der Korpus-Wahrscheinlichkeit) das Optimierungskriterium, während der EM-Algorithmus darauf abzielt, die EM-Hilfsfunktion (iterativ) zu maximieren. Diese unterscheidet sich lediglich um die sogenannte *erwartete Cross-Entropie aller Analysen* von der Log-Likelihood (Lemma, siehe unten).

3. Zusammenfassend erhält man das gewünschte Resultat, dass im EM-Algorithmus (iterativ) die empirische Erwartung

   $$\tilde{p}\,[\ p_\Phi(.|.)\,[\ \log\ p_\Theta\ ]\ ]$$

   zu maximieren ist, was (i) intuitiv annähernd das gleiche Ergebnis liefern könnte, wie die Maximierung der empirischen Erwartung

   $$\tilde{p}\,[\ \log\ p_\Theta\ ]$$

   in der Maximum-Likelihood-Estimierung, aber (ii) als Maximierung einer Summe von Logarithmen mathematisch sehr viel einfacher ist als die Maximierung des Logarithmus einer Summe.

Ein formaler Vergleich der Log-Likelihood des EM-Algorithmus mit der EM-Hilfsfunktion

Die nächsten Schritte haben das Ziel, den formalen Zusammenhang zwischen der Log-Likelihood des EM-Algorithmus und der EM-Hilfsfunktion aufzudecken. Die vorangegangene

Diskussion hat insbesondere gezeigt, dass die bedingten Wahrscheinlichkeitsverteilungen ($y \in \mathcal{Y}$):

$$p_\Phi(.|y) : \; X(y) \to [0,1] \; ,$$

wegen der Erwartungen

$$p_\Phi(.|y) \; [\; \log \; p_\Theta \;]$$

im engeren Zentrum des Interesses stehen. Gemäss Abschnitt 2.1 kann jede Wahrscheinlichkeitsverteilung als ein Korpus interpretiert werden. Entsprechend gilt dies auch für die bedingten Wahrscheinlichkeitsverteilung $p_\Phi(.|y)$, die für jeden unvollständigen Datentyp $y \in \mathcal{Y}$ als *gegenwärtiger Korpus der Analysen* $X(y)$ interpretiert werden könnten. In Abschnitt 2.3 wurden Entropie und Cross-Entropie als wichtige Bewertungsgrössen eines Korpus eingeführt. Angewandt auf den gegenwärtigen Korpus der Analysen $X(y)$ liefert dies die folgende Definition:

Definition (Entropie und Cross-Entropie der Analysen $X(y)$): Sei $y \in \mathcal{Y}$ ein unvollständiger Datentyp und $\Phi, \; \Theta \in \Omega$ zwei Parameter.

$$\begin{aligned} H(\; p_\Phi(.|y) \;) &= - \; p_\Phi(.|y) \; [\; \log \; p_\Phi(.|y) \;] \\ &= - \sum_{x \in X(y)} p_\Phi(x|y) \cdot \log \; p_\Phi(x|y) \end{aligned}$$

heisst *Entropie der Analysen* $X(y)$.

$$\begin{aligned} H_{\text{cross}}(\; p_\Theta(.|y); \; p_\Phi(.|y) \;) &= - \; p_\Phi(.|y) \; [\; \log \; p_\Theta(.|y) \;] \\ &= - \sum_{x \in X(y)} p_\Phi(x|y) \cdot \log \; p_\Theta(x|y) \end{aligned}$$

heisst *Cross-Entropie der Analysen* $X(y)$.

Während die Cross-Entropie der Analysen $X(y)$ die erwartete Anzahl von Datentoken in *bits* (logarithmische Basis 2) oder *nats* (Basis e) angibt, die von der Wahrscheinlichkeitsverteilung $p_\Theta(.|y)$ an einer beliebigen (aber festen) Position des Korpus $p_\Phi(.|y)$ generiert werden, muss die Entropie der Analysen $X(y)$ als deren untere Schranke angesehen werden.

Das wichtigste Ergebnis, welches in Abschnitt 2.1 vorgestellt (und in Abschnitt 2.4 bewiesen) wurde, aber auch aus der tiefen *Informations-Ungleichung der Informationstheorie* gefolgert werden kann, besagt, dass die Cross-Entropie eines Korpus stets grösser als seine Entropie ist. Angewandt auf die Analysen $X(y)$ ergibt dies das folgende Resultat:

Lemma (Entropie und Cross-Entropie der Analysen $X(y)$): Sei $y \in \mathcal{Y}$ ein unvollständiger Datentyp und $\Phi,\ \Theta \in \Omega$ zwei Parameter. Dann gilt:

$$H_{\text{cross}}(\ p_\Theta(.|y);\ p_\Phi(.|y)\ ) \geq H(\ p_\Phi(.|y)\ )\ .$$

Gleichheit gilt genau dann, wenn:

$$p_\Theta(.|y)\ =\ p_\Phi(.|y)\ .$$

Wenn auch völlig trivial, ist es doch ein Schlüsselschritt der folgenden Diskussion, dass sich dieses Ergebnis auf die empirische Erwartung überträgt.

Definition (Erwartete Entropie und Cross-Entropie aller Analysen):

$$\begin{aligned} H(\Phi) &= \tilde{p}\,[\ H(\ p_\Phi(.|.)\ )\ ] \\ &= \sum_{y\in\mathcal{Y}} \tilde{p}(y)\cdot H(\ p_\Phi(.|y)\ ) \\ &= -\sum_{y\in\mathcal{Y}} \tilde{p}(y) \sum_{x\in X(y)} p_\Phi(x|y)\cdot \log\ p_\Phi(x|y) \end{aligned}$$

heisst *erwartete Entropie aller Analysen.*

$$\begin{aligned} H_{\text{cross}}(\Theta;\Phi) &= \tilde{p}\,[\ H_{\text{cross}}(\ p_\Theta(.|.);\ p_\Phi(.|.)\ )\ ] \\ &= \sum_{y\in\mathcal{Y}} \tilde{p}(y)\cdot H_{\text{cross}}(\ p_\Theta(.|y);\ p_\Phi(.|y)\ ) \\ &= -\sum_{y\in\mathcal{Y}} \tilde{p}(y) \sum_{x\in X(y)} p_\Phi(x|y)\cdot \log\ p_\Theta(x|y) \end{aligned}$$

heisst *erwartete Cross-Entropie aller Analysen.*

Damit folgt, wie obgen schon angekündigt:

Lemma (Erwartete Entropie und Cross-Entropie aller Analysen): Seien $\Phi,\ \Theta \in \Omega$ zwei Parameter. Dann gilt:

$$H_{\text{cross}}(\Theta;\ \Phi) \geq H(\Phi)\ .$$

Gleichheit gilt genau dann, wenn:

$$p_\Theta(.|y)\ =\ p_\Phi(.|y) \qquad \text{für alle } y \in \mathcal{Y} \text{ mit } \tilde{p}(y) > 0\ .$$

Beweis: Gemäss Definition gilt:

$$\begin{aligned} &H_{\text{cross}}(\Theta;\ \Phi) - H(\Phi) \\ &= \sum_{y\in\mathcal{Y}} \tilde{p}(y)\cdot (H_{\text{cross}}(\ p_\Theta(.|y); p_\Phi(.|y)\ ) - H(\ p_\Phi(.|y)\ )) \end{aligned}$$

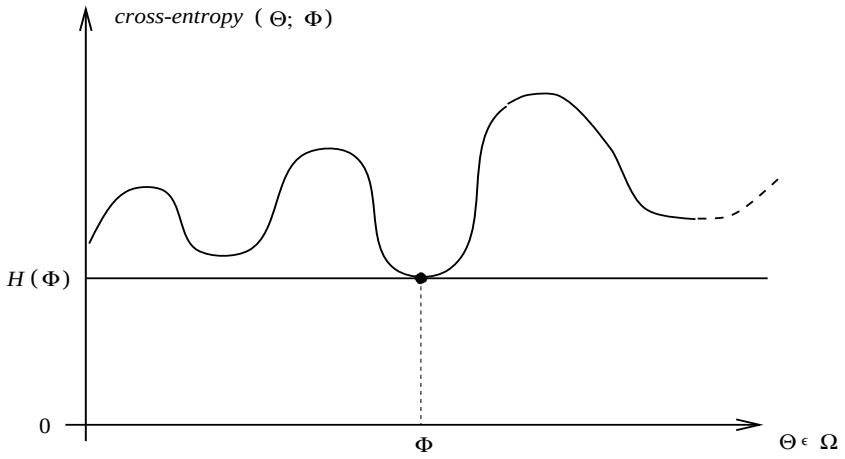


Abbildung 3.17: Erwartete Entropie und Cross-Entropie aller Analysen

Aus $\tilde{p}(y) \geq 0$ und dem ersten Lemma folgt, dass jeder Summand dieser Summe nicht-negativ ist. Hieraus ergibt sich der erste Teil der Behauptung. Ferner gilt, dass diese Summe genau dann Null ist, wenn alle ihre Summanden Null sind. Wieder mit dem ersten Lemma folgt hieraus der zweite Teil der Behauptung **q.e.d.**

Abbildung 3.17 skiziert den eben angegebenen Zusammenhang zwischen erwarteter Entropie und Cross-Entropie aller Analysen. Auf der x-Achse werden die Parameter $\Theta \in \Omega$ abgetragen, auf der y-Achse die Werte der erwarteten Cross-Entropie $H_{\text{cross}}(\Theta; \Phi)$. Die erwartete Entropie $H(\Phi)$ wurde als Konstante abgebildet, weil sie nicht von dem Parameter $\Theta \in \Omega$ abhängig ist. Da die Terme $\log\ p_\Theta(y)$ und $\log\ p_\Phi(y)$ kleiner oder gleich 0 sind, sind die Entropie und die Cross-Entropie der Analysen $X(y)$ nicht negativ. Dies überträgt sich auch auf ihre empirischen Erwartungswerte $H(\Phi)$ und $H_{\text{cross}}(\Theta; \Phi)$. Das Lemma besagt, dass die erwartete Cross-Entropie oberhalb der erwarteten Entropie liegt und wenigstens für den Parameter $\Theta = \Phi$ mit ihr übereinstimmt. Haben die parametrisierten Wahrscheinlichkeitsverteilungen $p_\Theta(.)$ und die symbolische Analyse-Komponente $X(.)$ die folgende Eigenschaft:

$$p_\Theta(.|y) = p_\Phi(.|y) \text{ für alle } y \in \mathcal{Y} \text{ mit } \tilde{p}(y) > 0 \quad \Rightarrow \quad \Theta = \Phi\ ,$$

so ist $\Theta = \Phi$ tatsächlich auch der einzige Punkt, an dem die erwartete Cross-Entropie mit der erwarteten Entropie übereinstimmen kann. Da es unklar ist, ob man für den EM-Algorithmus fordern sollte, dass dessen Parametrisierung und Analyse-Komponente die angegebene Eigenschaft besitzen, weiss man nicht, ob es weitere Paramter Θ gibt, an denen die erwartete Cross-Entropie mit der erwarteten Entropie übereinstimmt. In Abbildung 3.17 wurde deshalb nur der eine Punkt $\Theta = \Phi$ eingezeichnet, von dem man dies sicher weiss.

Das nächste (sehr schöne) Resultat zeigt, dass sich die Log-Likelihood des EM-Algorith-

mus und die EM-Hilfsfunktion gerade um die erwartete Cross-Entropie aller Analysen unterscheiden.

Lemma (Log-Likelihood und EM-Hilfsfunktion): Seien $\Phi,\ \Theta \in \Omega$ zwei Parameter. Dann gilt:

$$L(\Theta) \ = \ Q_{\Phi}(\Theta) \ + \ H_{\text{cross}}(\Theta;\ \Phi) \ .$$

Beweis: Für alle $y \in \mathcal{Y}$ und alle $\Phi \in \Omega$ gilt $\sum_{x \in X(y)} p_{\Phi}(x|y) = 1$. Daher erhält man für die Log-Likelihood:

$$\begin{aligned} L(\Theta) &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \cdot \log p_{\Theta}(y) \\ &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \cdot \log p_{\Theta}(y) \sum_{x \in X(y)} p_{\Phi}(x|y) \\ &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in X(y)} p_{\Phi}(x|y) \cdot \log p_{\Theta}(y) \end{aligned}$$

Da für jede Analyse $x \in X(y)$ gilt, dass:

$$p_{\Theta}(y) = \frac{p_{\Theta}(x)}{p_{\Theta}(x|y)} \ ,$$

folgt weiter:

$$\begin{aligned} L(\Theta) &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in X(y)} p_{\Phi}(x|y) \cdot \log \frac{p_{\Theta}(x)}{p_{\Theta}(x|y)} \\ &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in X(y)} p_{\Phi}(x|y) \cdot (\log p_{\Theta}(x) - \log p_{\Theta}(x|y)) \\ &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in X(y)} p_{\Phi}(x|y) \cdot \log p_{\Theta}(x) \\ &\quad - \sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in X(y)} p_{\Phi}(x|y) \cdot \log p_{\Theta}(x|y) \\ &= Q_{\Phi}(\Theta) + H_{\text{cross}}(\Theta;\Phi) \qquad \textbf{q.e.d.} \end{aligned}$$

Abbildung 3.18 greift eine Idee aus Abschnitt 3.3 auf und zeigt die Cross-Entropie im Kontext wichtiger stochastischer Komponenten des EM-Algorithmus (vergleiche auch Abbildung 3.9). Die Abbildung ist so organisiert, dass diejenigen Komponenten des EM-Algorithmus, die unvollständige Datentypen beschreiben, im linken Teil der Abbildung gezeigt werden, während die Komponenten, die vollständige Datentypen beschreiben, im rechten Teil der Abbildung zu finden sind. Somit muss die empirische Wahrscheinlichkeitsverteilung $\tilde{p}(y)$ und die auf ihr berechnete Log-Likelihood $L(\Theta)$ links, und die empirische Wahrscheinlichkeitsverteilung $\tilde{p}_{\Phi}(x)$ und die auf ihr berechnete Log-Likelihood $L_{\Phi}(\Theta)$ rechts

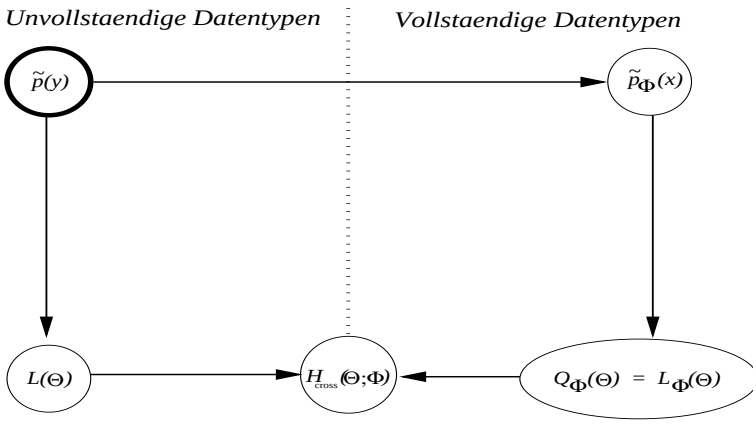


Abbildung 3.18: Die erwartete Cross-Entropie im Kontext der stochastischen Komponenten des EM-Algorithmus

dargestellt werden. Die Cross-Entropie, die sich als Differenz der beiden Log-Likelihood-Funktionen auf unvollständigen bzw. vollständigen Datentypen darstellen lässt, kann daher weder dem linken noch dem rechten Teil der Abbildung zugeordnet werden und wird deshalb in der Mitte dieser beiden wichtigen Funktionen angesiedelt.

Es folgt eine interessante Bemerkung, die darauf abzielt, die neu gewonnene Erkenntnis zur Bestimmung optimaler Modelle zu nutzen.

Korollar: Für alle Parameter $\Theta, \Phi \in \Omega$ gilt:

$$L(\Theta) \;\geq\; Q_\Phi(\Theta) \;+\; H(\Phi)\;.$$

Somit wird bei der Maximierung der EM-Hilfsfunktion zumindest eine untere Schranke der Log-Likelihood des EM-Algorithmus maximiert.

Beweis: Folgt aus dem letzten Lemma und dem Lemma über die erwartete Cross-Entropie und Entropie. **q.e.d.**

Mathematische Grundlagen aus der Differentialrechnung für den EM-Algorithmus

Die vorangegangene Diskussion hat gezeigt, dass die Maximum-Likelihood-Estimierung mit der globalen Maximierung der Log-Likelihood und der EM-Algorithmus mit der iterativen Maximierung der EM-Hilfsfunktionen beschäftigt ist. Obwohl motiviert wurde, dass die Maximierung der EM-Hilfsfunktion die leichtere von beiden Aufgaben ist, handelt es sich bei der EM-Hilfsfunktion (wie bei der Log-Likelihood) im allgemeinen um eine Funktion mit mehrdimensionalen Parameter-Argumenten und es ist bekannt, dass

die Maximierung solcher Funktionen mathematisch anspruchsvoll ist.

In diesem Abschnitt sollen deshalb die wichtigsten Begriffe und Sätze aus der Differentialrechnung vorgestellt werden, die im weiteren Verlauf von Nutzen sein könnten. Einleitend wurde bereits darauf hingewiesen, dass die *globale Maximierung* einer Funktion ein dreistufiger Prozess ist: (i) der *lokalen Maximierung* folgt (ii) die Bestimmung der *Randwerte* und (iii) die Auswahl der *globalen Maxima* aus der Menge der lokalen Maxima und Randwerte (vergleiche Abbildung 3.15).

Zur Bestimmung der lokalen Maxima gibt es eine Reihe von mathematischen Theoremen, in denen die *partiellen ersten und zweiten Ableitungen* eine wichtige Rolle spielen:

Definition: Für $\Theta = < t_1 \dots t_d > \in \Omega \subseteq \mathbb{R}^d$ wird definiert:

- Die *erste partielle Ableitung nach* t_r wird mit ∂_r bezeichnet. D.h. es gilt:

$$\partial_r = \frac{\partial}{\partial t_r} \ .$$

- Der *Gradient* ∇_Θ ist der *Vektor aller ersten partiellen Ableitungen.* D.h. es gilt:

$$\nabla_\Theta \ = < \partial_r \mid r = 1 \dots d > \ .$$

- Die *zweite partielle Ableitung nach* t_r *und nach* t_s wird mit $\partial^2_{r,s}$ bezeichnet. D.h. es gilt:

$$\partial^2_{r,s} \ = \ \partial_r \ \partial_s \ = \ \frac{\partial^2}{\partial t_r \partial t_s} \ .$$

- Die *Matrix aller zweiten partiellen Ableitungen* wird mit ∂^2_Θ bezeichnet. D.h. es gilt:

$$\partial^2_\Theta \ = \ \left(\partial^2_{r,s}\right)_{r,s=1\dots d} \ = \ \left(\frac{\partial^2}{\partial t_r \partial t_s}\right)_{r,s=1\dots d} \ .$$

Der folgende Satz beschreibt eine wichtige Beobachtung:

Satz 61 (Erwe (1962), Seite 344): Wenn die Funktion $F(\Theta)$ partiell differenzierbar ist und in Θ^* ein *relatives Extremum* (Maximum oder Minimum) besitzt, so gilt:

$$\nabla_\Theta F(\Theta)|_{\Theta=\Theta^*} = 0 \ .$$

Leider gilt die Umkehrung nicht. Abbildung 3.19 zeigt ein Gegenbeispiel. Während für den Parameter Φ tatsächlich ein relatives Extremum vorliegt, ist dies für Φ' nicht der Fall, obwohl in beiden Fällen:

$$\nabla_\Theta F(\Theta)|_{\Theta=\Phi} = \nabla_\Theta F(\Theta)|_{\Theta=\Phi'} = 0 \ .$$

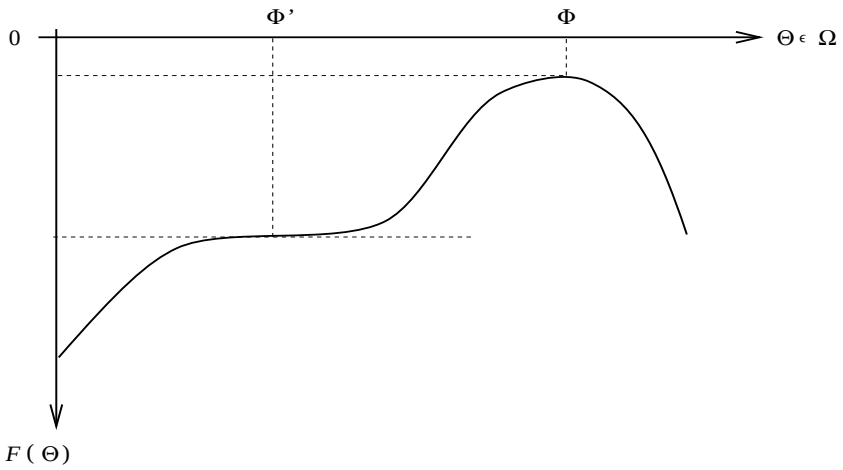


Abbildung 3.19: Maximum-Likelihood-Estimierung

Definition: Ein Parameter $\Theta^* \in \Omega$ heisst *stationärer Punkt der Funktion* $F(\Theta)$, falls der *Gradient von* $F(\Theta)$ an dieser Stelle Null ist:

$$\nabla_\Theta F(\Theta)|_{\Theta=\Theta^*} = 0 \ .$$

Ein Wert $F^* \in \mathbb{R}$ heisst *stationärer Wert der Funktion* F, falls

$$F^* = F(\Theta^*)$$

für einen stationären Punkt $\Theta^* \in \Omega$.

Satz 61 (Erwe 1962) besagt, dass an einem stationären Punkt möglicherweise ein *relativer Extremwert (Maximum oder Minimum) der Funktion* F vorliegt. Abbildung 3.19 zeigt jedoch, dass eventuell auch ein sogenannter *Sattelpunkt* (in keinem Fall aber ein beliebiger Funktionswert) vorliegen kann. Der folgende Satz gibt hinreichende Bedingungen für *relative Extrema* an.

Satz 62 (Erwe (1962), Seite 345): Sei die Funktion $F(\Theta)$ zweimal stetig differenzierbar ist. Wenn

$$\nabla_\Theta F(\Theta)|_{\Theta=\Theta^*} = 0 \ ,$$

und

$$\partial^2_\Theta F(\Theta)\Big|_{\Theta=\Theta^*}$$

negativ definit (bzw. *positiv definit*) ist, so ist Θ^* ein strenges *relatives Maximum* (bzw. *relatives Minimum*).

Die Begriffe *positiv/negativ definit* und der damit zusammenhängenden Begriff des *Eigenwerts* wurden bereits in Kapitel 2 eingeführt.

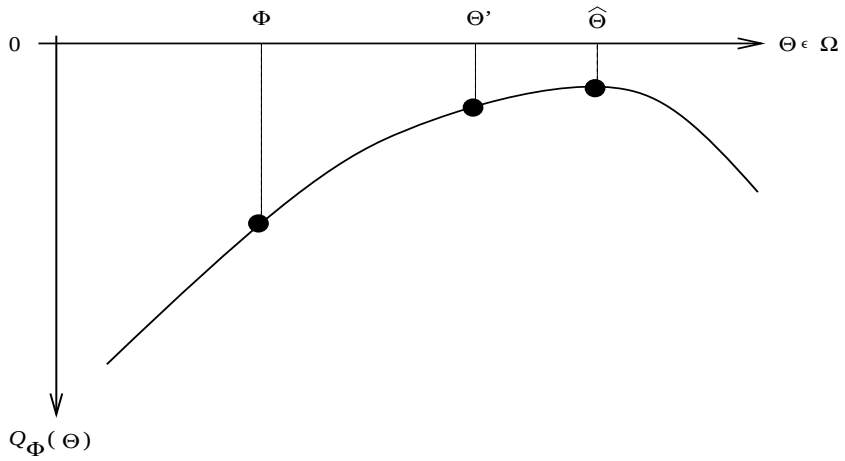


Abbildung 3.20: GEM-Algorithmus und EM-Algorithmus

Einführung des EM-Algorithmus gemäss der originalen Arbeit von Dempster, Laird und Rubin (1977)

Es ist interessant, dass seit der Einführung des EM-Algorithmus durch Dempster et al. (1977) alle wichtigen Aussagen über den EM-Algorithmus sogar für den sogenannten *generalisierten EM-Algorithmus* bewiesen wurden.

Definition (Dempster et al. 1977): Statt der EM-Parameter-Abbildung:

$$\mathcal{M}: \ \Omega \to \Omega, \quad \Phi \mapsto \text{argmax}_{\Theta \in \Omega} \ Q_\Phi(\Theta) \ ,$$

die fordert, dass:

$$Q_\Phi(\mathcal{M}(\Phi)) \geq Q_\Phi(\Theta) \qquad \text{für alle } \Phi, \ \Theta \in \Omega \ ,$$

wird in einem *generalisierten EM-Algorithmus (GEM-Algorithmus)* von der Parameter-Abbildung $\mathcal{M}(.)$ lediglich verlangt, dass:

$$Q_\Phi(\mathcal{M}(\Phi)) \geq Q_\Phi(\Phi) \qquad \text{für alle } \Phi \in \Omega \ .$$

Abbildung 3.20 skizziert den Unterschied zwischen einen GEM-Algorithmus und einen EM-Algorithmus. Auf der x-Achse werden die Parameter $\Theta \in \Omega$ abgetragen, auf der y-Achse die Werte der EM-Hilfsfunktion $Q_\Phi(\Theta)$. Offensichtlich kann die EM-Hilfsfunktion nur negative Werte anehmen. Obwohl in der Abbildung eine *konkave* EM-Hilfsfunktion gezeigt wird, deren globales Maximum $\hat{\Theta}$ ein lokales Maximum ist, ist es sehr zweifelhaft (obschon wünschenswert), dass beliebige EM-Hilfsfunktionen diese Eigenschaft besitzen. Der Unterschied zwischen einem GEM-Algorithmus und einem EM-Algorithmus liegt in

deren Parameter-Abbildungen. Ein EM-Algorithmus verlangt, dass der gegenwärtige Parameter Φ auf das globale Maximum $\hat{\Theta}$ der EM-Hilfsfunktion $Q_\Phi(.)$ abgebildet wird:

$$\mathcal{M}(\Phi) = \hat{\Theta} \ .$$

Ein GEM-Algorithmus fordert lediglich, dass der gegenwärtige Parameter Φ auf einen Parameter Θ' abgebildet wird, der zu einem grösseren Wert der EM-Hilfsfunktion $Q_\Phi(.)$ führt:

$$\mathcal{M}(\Phi) = \Theta' \qquad \text{mit } Q_\Phi(\Theta') \geq Q_\Phi(\Phi) \ .$$

Offensichtlich ist daher jeder EM-Algorithmus ein GEM-Algorithmus. In der Arbeit von Dempster et al. (1977) finden sich vier Theoreme zum GEM-Algorithmus, die alle die folgenden Annahmen machen, ohne sie jeweils explizit aufzuführen.

Voraussetzungen (Dempster et al. 1977):

- $\Omega \subseteq \mathbb{R}^d$, d.h. der Parameterraum ist eine Teilmenge eines mehrdimensionalen euklidischen Raumes,
- Die Log-Likelihood L, die EM-Hilfsfunktionen Q_Φ, die erwartete Cross-Entropie H_{cross} aller Analysen und die EM-Parameter-Abbildung $\mathcal{M}$ sind "genügend oft" *stetig differenzierbar*,
- Differenzierung und Summation sind vertauschbar.

Obwohl zwei der genannten vier Theoreme (zur sogenannten *Parameter-Konvergenz*) relativ rasch durch Gegenbeispiele (Boyles 1983) widerlegt wurden, erwies sich das folgende Theorem von fundamentaler Bedeutung für die EM-Theorie. Hierin wird gezeigt, dass eine iterative Anwendung der EM-Parameter-Abbildung in eine Folge von Parametern resultiert, deren Log-Likelihood-Werte monoton wachsen (siehe Abbildung 3.7).

Theorem (Theorem 1 aus Dempster et al. (1977)): In einem GEM-Algorithmus gilt:

$$L(\mathcal{M}(\Phi)) \geq L(\Phi) \qquad \text{für alle } \Phi \in \Omega \ .$$

Gleichheit liegt genau dann vor, wenn die folgenden beiden Gleichungen erfüllt sind:

(i) $Q_\Phi(\mathcal{M}(\Phi)) = Q_\Phi(\Phi)$

(ii) $p_{\mathcal{M}(\Phi)}(.|y) = p_\Phi(.|y)$ für alle $y \in \mathcal{Y}$ mit $\tilde{p}(y) > 0$.

Beweis: Mit dem Lemma über die Log-Likelihood und die EM-Hilfsfunktion folgt:

$$L(\mathcal{M}(\Phi)) - L(\Phi) \quad = \quad Q_\Phi(\mathcal{M}(\Phi)) + H_{\text{cross}}(\mathcal{M}(\Phi);\ \Phi)$$

$$
\begin{aligned}
& - Q_\Phi(\Phi) - H_{\text{cross}}(\Phi;\ \Phi) \\
= \ & Q_\Phi(\mathcal{M}(\Phi)) - Q_\Phi(\Phi) \\
& + H_{\text{cross}}(\mathcal{M}(\Phi);\ \Phi) - H(\Phi) \ .
\end{aligned}
$$

Für jeden GEM-Algorithmus gilt:

$$Q_\Phi(\mathcal{M}(\Phi)) - Q_\Phi(\Phi) \geq 0 \ .$$

Gemäss dem Lemma über die erwartete Cross-Entropie und Entropie gilt:

$$H_{\text{cross}}(\mathcal{M}(\Phi);\ \Phi) - H(\Phi) \geq 0 \ .$$

Es folgt $L(\mathcal{M}(\Phi)) - L(\Phi) \geq 0$. Gleichheit tritt genau dann ein, wenn sowohl

$$Q_\Phi(\mathcal{M}(\Phi)) - Q_\Phi(\Phi) = 0 \ ,$$

als auch $H_{\text{cross}}(\mathcal{M}(\Phi);\ \Phi) - H(\Phi) = 0$. Gemäss dem Lemma über die erwartete Cross-Entropie und Entropie der Analysen tritt das letztere genau dann ein, wenn

$$p_{\mathcal{M}(\Phi)}(.|y) = p_\Phi(.|y) \quad \text{für alle } y \in \mathcal{Y} \text{ mit } \tilde{p}(y) > 0 \qquad \textbf{q.e.d.}$$

Es ist äusserst interessant, dass Dempster et al. (1977) in einer Randnote bemerken, dass das Lemma über die erwartete Cross-Entropie und Entropie der Analysen, sowie das Theorem 1 bereits von Baum et al. (1970) für den Spezialfall der Hidden-Markov-Modelle vorgestellt und bewiesen wurden.

Die nächsten zwei Korollare geben notwendige (jedoch nicht hinreichende) Bedingungen dafür an, dass eine (iterative) Anwendung der EM-Parameter-Abbildung eine Maximum-Likelihood-Estimierung $\hat{\Theta}$ ausgibt.

Korollar (Korollar 1 aus Dempster et al. (1977)): Der mit einem GEM-Algorithmus re-estimierte Parameter $\mathcal{M}(\hat{\Theta})$ einer Maximum-Likelihood-Estimierung $\hat{\Theta}$ (d.h. $L(\hat{\Theta}) \geq L(\Theta)$ für alle $\Theta \in \Omega$) hat die folgenden drei Eigenschaften:
(i) die Log-Likelihood ist *invariant*:

$$L(\mathcal{M}(\hat{\Theta})) = L(\hat{\Theta}) \ ,$$

(ii) die EM-Hilfsfunktion ist *invariant*:

$$Q_{\hat{\Theta}}(\mathcal{M}(\hat{\Theta})) = Q_{\hat{\Theta}}(\hat{\Theta})$$

(iii) die bedingten Wahrscheinlichkeitsverteilungen über die Analysen $X(y)$ sind *invariant*:

$$p_{\mathcal{M}(\hat{\Theta})}(.|y) = p_{\hat{\Theta}}(.|y) \quad \text{für alle } y \in \mathcal{Y} \text{ mit } \tilde{p}(y) > 0 \ .$$

Beweis: Das Theorem 1 aus Dempster et al. (1977) liefert:

$$L(\mathcal{M}(\hat{\Theta})) \geq L(\hat{\Theta}) \ .$$

Da $\hat{\Theta}$ eine Maximum-Likelihood-Estimierung ist, folgt andererseits auch:

$$L(\hat{\Theta}) \geq L(\mathcal{M}(\hat{\Theta})) \ .$$

Damit ist Eigenschaft (i) bewiesen, aus der mit Theorem 1 aus Dempster et al. (1977) die Eigenschaften (ii) und (iii) folgen. **q.e.d.**

Korollar (Korollar 2 aus Dempster et al. (1977)): Der mit einem GEM-Algorithmus re-estimierte Parameter $\mathcal{M}(\hat{\Theta})$ einer eindeutigen Maximum-Likelihood-Estimierung $\hat{\Theta}$ (d.h. $L(\hat{\Theta}) > L(\Theta)$ für alle $\Theta \in \Omega \setminus \{\hat{\Theta}\}$) ist *Parameter-invariant*:

$$\mathcal{M}(\hat{\Theta}) = \hat{\Theta} \ .$$

Beweis: Korollar 1 aus Dempster et al. (1977) liefert: $L(\mathcal{M}(\hat{\Theta})) = L(\hat{\Theta})$. Da $\hat{\Theta}$ eine eindeutige Maximum-Likelihood-Estimierung ist, folgt die Behauptung. **q.e.d.**

Das folgende Lemma enthälte eine für die EM-Theorie grundlegende Beobachtung.

Lemma (Lemma 2 aus Dempster et al. (1977)): Sei $\Phi \in \Omega$. (i) Es gilt:

$$\nabla_\Theta \ H_{\text{cross}}(\Theta, \Phi)|_{\Theta=\Phi} \ = \ 0 \ .$$

D.h. die erwartete Cross-Entropie aller Analysen hat an der Stelle $\Theta = \Phi$ einen *stationären Punkt*. (ii) Ist die Matrix $\partial^2_\Theta \ H_{\text{cross}}(\Theta, \Phi)|_{\Theta=\Phi}$ *positiv definit*, so hat die erwartete Cross-Entropie aller Analysen an der Stelle $\Theta = \Phi$ ein *strenges lokales Minimum*. Unter Benutzung von $\Delta_r p_\Theta(x|y) = \partial_r \log \ p_\Theta(x|y)$ gilt:

$$\begin{aligned} \partial^2_{r,s} H_{\text{cross}}(\Theta, \Phi)|_{\Theta=\Phi} &= \tilde{p}\,[\ p_\Phi(.|.)\,[\ \Delta_r p_\Theta(.|.)|_{\Theta=\Phi} \cdot \Delta_s p_\Theta(.|.)|_{\Theta=\Phi}\ ]\ ] \\ &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in X(y)} p_\Phi(x|y) \cdot \Delta_r p_\Theta(x|y)|_{\Theta=\Phi} \cdot \Delta_s p_\Theta(x|y)|_{\Theta=\Phi} \ . \end{aligned}$$

Beweis: Erster Beweis zu (i): Dem Lemma über die erwartete Entropie und Cross-Entropie aller Analysen entnimmt man, dass $H_{\text{cross}}(\Theta, \Phi)$ als Funktion von Θ an der Stelle $\Theta = \Phi$ ein relatives Minimum besitzt. Aus der Differential- und Integralrechnung ist bekannt (Satz 61, Erwe (1962)), dass die relativen Extrema (Minima oder Maxima) von differenzierbaren Funktionen stationäre Punkte sein müssen[8].

[8] Gemäss Satz 63 (Erwe 1962) folgt zusätzlich, dass $\partial^2_\Theta \ H_{\text{cross}}(\Theta, \Phi)|_{\Theta=\Phi}$ *positiv semidefinit* ist.

Zweiter Beweis zu (i) (Dempster et al. (1977)): Per Definition gilt:

$$H_{\text{cross}}(\Theta;\Phi) = -\sum_{y\in\mathcal{Y}} \tilde{p}(y) \sum_{x\in X(y)} p_\Phi(x|y)\cdot \log\ p_\Theta(x|y)$$

Es folgt:

$$\begin{aligned}\partial_r H_{\text{cross}}(\Theta;\Phi) &= -\sum_{y\in\mathcal{Y}} \tilde{p}(y) \sum_{x\in X(y)} p_\Phi(x|y)\cdot \partial_r \log\ p_\Theta(x|y)\\ &= -\sum_{y\in\mathcal{Y}} \tilde{p}(y) \sum_{x\in X(y)} p_\Phi(x|y)\cdot \frac{\partial_r p_\Theta(x|y)}{p_\Theta(x|y)}\ .\end{aligned}$$

Somit:

$$\begin{aligned}\partial_r H_{\text{cross}}(\Theta;\Phi)|_{\Theta=\Phi} &= -\sum_{y\in\mathcal{Y}} \tilde{p}(y) \sum_{x\in X(y)} p_\Phi(x|y)\cdot \frac{\partial_r p_\Theta(x|y)|_{\Theta=\Phi}}{p_\Phi(x|y)}\\ &= -\sum_{y\in\mathcal{Y}} \tilde{p}(y) \sum_{x\in X(y)} p_\Phi(x|y)\cdot \frac{\partial_r p_\Theta(x|y)|_{\Theta=\Phi}}{p_\Phi(x|y)}\\ &= -\sum_{y\in\mathcal{Y}} \tilde{p}(y) \sum_{x\in X(y)} \partial_r p_\Theta(x|y)|_{\Theta=\Phi}\\ &= -\sum_{y\in\mathcal{Y}} \tilde{p}(y)\cdot \partial_r \left(\sum_{x\in X(y)} p_\Theta(x|y)\right)\Bigg|_{\Theta=\Phi}\\ &= 0\ ,\end{aligned}$$

denn die Summe $\sum_{x\in X(y)} p_\Theta(x|y)$ ist für alle Parameter $\Theta\in\Omega$ konstant 1, hat also nur erste partielle Ableitungen, die gleich 0 sind. Hieraus folgt die Behauptung.

zu (ii): Aus dem im Beweis zu (i) hergeleiteten Term für $\partial_r H_{\text{cross}}(\Theta;\Phi)$ folgt durch erneute Differentierung:

$$\partial_s\partial_r H_{\text{cross}}(\Theta;\Phi) = -\sum_{y\in\mathcal{Y}} \tilde{p}(y) \sum_{x\in X(y)} p_\Phi(x|y)\cdot \partial_s\left(\frac{\partial_r p_\Theta(x|y)}{p_\Theta(x|y)}\right)$$

Unter Benutzung der *Quotientenregel der Differentialrechnung* folgt:

$$\begin{aligned}\partial_s\left(\frac{\partial_r p_\Theta(x|y)}{p_\Theta(x|y)}\right) &= \frac{(\partial_s\partial_r p_\Theta(x|y))\cdot p_\Theta(x|y) - (\partial_r p_\Theta(x|y))\cdot(\partial_s p_\Theta(x|y))}{p_\Theta^2(x|y)}\\ &= \frac{\partial_s\partial_r p_\Theta(x|y)}{p_\Theta(x|y)} - \frac{\partial_r p_\Theta(x|y)}{p_\Theta(x|y)}\cdot\frac{\partial_s p_\Theta(x|y)}{p_\Theta(x|y)}\\ &= \frac{\partial_s\partial_r p_\Theta(x|y)}{p_\Theta(x|y)} - \Delta_r p_\Theta(x|y)\cdot\Delta_s p_\Theta(x|y)\end{aligned}$$

Somit:

$$\begin{aligned}\partial_s\partial_r H_{\text{cross}}(\Theta;\Phi) &= -\sum_{y\in\mathcal{Y}} \tilde{p}(y) \sum_{x\in X(y)} p_\Phi(x|y)\cdot\frac{\partial_s\partial_r p_\Theta(x|y)}{p_\Theta(x|y)}\\ &\quad + \sum_{y\in\mathcal{Y}} \tilde{p}(y) \sum_{x\in X(y)} p_\Phi(x|y)\cdot\Delta_r p_\Theta(x|y)\cdot\Delta_s p_\Theta(x|y)\ .\end{aligned}$$

Es folgt:

$$\begin{aligned}\partial^2_{r,s} H_{\text{cross}}(\Theta;\Phi)\Big|_{\Theta=\Phi} &= -\sum_{y\in\mathcal{Y}} \tilde{p}(y) \sum_{x\in X(y)} \partial_s \partial_r p_\Theta(x|y)|_{\Theta=\Phi} \\ &\quad + \sum_{y\in\mathcal{Y}} \tilde{p}(y) \sum_{x\in X(y)} p_\Phi(x|y) \cdot \Delta_r p_\Theta(x|y)|_{\Theta=\Phi} \cdot \Delta_s p_\Theta(x|y)|_{\Theta=\Phi} \ .\end{aligned}$$

Während die erste Summe verschwindet (wobei man das selbe Argument benutzen kann, wie im Beweis zu Teil (ii)), ist die zweite Summe das gewünschte Ergebnis. **q.e.d.**

Der Sachverhalt wird in Abbildung 3.17 wiedergegeben. Es ist unklar, ob die erwartete Cross-Entropie der Analysen ausser an der Stelle $\Theta = \Phi$ noch weitere stationäre Punkte besitzt (was aber vermutet werden könnte). Der erste Teil dieses Lemmas ist der Schlüssel zu Theorem 4 (Dempster et al. 1977).

Die Theoreme 2 und 3 (Dempster et al. 1977) adressierten die Frage, ob die von einem GEM-Algorithmus ausgegebene Parameterfolge konvergiert. Leider wurden beide Theoreme widerlegt. Das folgende Theorem zeigt lediglich, dass eine konvergente Parameterfolge (i) einen *stationären Punkt der Log-Likelihood* als Grenzwert Θ^* besitzen muss (vorausgesetzt die Parameterfolge besteht aus stationären Punkten der EM-Hilfsfunktion, was wohl in der Praxis stets der Fall sein wird) und (ii) eine *Konvergenzrate* $\nabla_\Theta \mathcal{M}(\Theta)|_{\Theta=\Theta^*}$ aufweisen wird, die sich mit genau zwei seperaten Informationen über unvollständige bzw. vollständige Datentypen berechnen lässt.

Theorem (Theorem 4 aus Dempster et al. (1977)): (i) Sei

$$\Theta_0,\ \Theta_1,\ \Theta_2,\ \Theta_3,\ \ldots$$

die Parameterfolge eines GEM-Algorithmus, sodass:

- Die Parameterfolge konvergiert:

$$\Theta_i \to \Theta^* \qquad (i \to \infty)\ .$$

- Der Parameter Θ_i ist ein *stationärer Punkt* der EM-Hilfsfunktion $Q_{\Theta_{i-1}}$, d.h.

$$\nabla_\Theta Q_{\Theta_{i-1}}(\Theta)\big|_{\Theta=\Theta_i} \ = \ 0\ .$$

Dann gilt:

- Der Grenzwert Θ^* ist ein *stationärer Punkt* der Log-Likelihood, d.h.

$$\nabla_\Theta L(\Theta)|_{\Theta=\Theta^*} = 0\ .$$

(ii) Gilt ferner, dass die *Eigenwerte* der *Matrix der zweiten partiellen Ableitungen* der EM-Hilfsfunktion $Q_{\Theta_{i-1}}$ an der Stelle Θ_i

$$\partial^2_\Theta Q_{\Theta_{i-1}}(\Theta)\Big|_{\Theta=\Theta_i}$$

kleiner oder gleich einem (für alle $i = 1, 2, 3, \ldots$ fixierten) $\lambda < 0$ sind, so folgt nicht nur, dass die EM-Hilfsfunktionen $Q_{\Theta_{i-1}}$ an der Stelle Θ_i *negativ definit* sind, sondern auch, dass die *Matrix der partiellen zweiten Ableitungen* der EM-Hilfsfunktion Q_{Θ^*} an der Stelle Θ^*

$$\partial^2_\Theta Q_{\Theta^*}(\Theta)\Big|_{\Theta=\Theta^*}$$

negativ definit ist, sodass für die Parameter-Abbildung des GEM-Algorithmus an der Stelle Θ^* gilt, dass

$$\nabla_\Theta \mathcal{M}(\Theta)|_{\Theta=\Theta^*} = \partial^2_\Theta H_{\text{cross}}(\Theta;\ \Theta^*)\Big|_{\Theta=\Theta^*} \ \cdot \ \left(\partial^2_\Theta Q_{\Theta^*}(\Theta)\Big|_{\Theta=\Theta^*}\right)^{-1}$$

Beweis: Für den Beweis von (ii) wird auf Dempster et al. (1977) verwiesen, weil im weiteren Verlauf dieser Arbeit nur der erste Teil des Theorems benutzt wird. Zu (i): Mit dem Lemma über die Log-Likelihood und die EM-Hilfsfunktion folgt durch partielle Differenzierung:

$$\nabla_\Theta L(\Theta) \ = \ \nabla_\Theta Q_\Phi(\Theta) + \nabla_\Theta H_{\text{cross}}(\Theta;\Phi)\ .$$

Es folgt für $\Phi = \Theta_{i-1}$, dass an der Stelle $\Theta = \Theta_i$ gilt:

$$\nabla_\Theta L(\Theta)|_{\Theta=\Theta_i} \;=\; \nabla_\Theta Q_{\Theta_{i-1}}(\Theta)\big|_{\Theta=\Theta_i} + \nabla_\Theta \; H_{\text{cross}}(\Theta, \Theta_{i-1})|_{\Theta=\Theta_i} \; .$$

Wegen der zweiten Voraussetzung an die Parameterfolge wird dies zu:

$$\nabla_\Theta L(\Theta)|_{\Theta=\Theta_i} \;=\; \nabla_\Theta \; H_{\text{cross}}(\Theta, \Theta_{i-1})|_{\Theta=\Theta_i} \; .$$

Grenzübergang $i \to \infty$ liefert wegen der ersten Voraussetzung an die Parameterfolge, dass:

$$\nabla_\Theta L(\Theta)|_{\Theta=\Theta^*} \;=\; \nabla_\Theta \; H_{\text{cross}}(\Theta, \Theta^*)|_{\Theta=\Theta^*} \; .$$

Mit dem vorangegangenen Lemma folgt hieraus die Behauptung:

$$\nabla_\Theta L(\Theta)|_{\Theta=\Theta^*} \;=\; 0 \; .$$

q.e.d.

Verbesserte Aussagen zum Konvergenzverhalten des EM-Algorithmus durch Wu (1983)

Die Arbeit von Wu (1983) wurde durch die folgende Ausgangslage motiviert:

- Obwohl es die ursprüngliche Absicht bei Einführung des EM-Algorithmus war, eine iterative Berechnungsmethode für die Maximum-Likelihood-Estimierung zu etablieren, war es kurze Zeit später bekannt, dass der EM-Algorithmus nicht unter allen Umständen garantiert, dass

  $$L^* = \lim_{i\to\infty} L(\Theta_i)$$

  ein *globales Maximum* oder zumindest ein *lokales Maximum* der Log-Likelihood ergibt. Vielmehr zeigte der EM-Algorithmus für viele Anwendungen das Verhalten, je nach gewähltem *Startparameter* in ein lokales Maximum oder in einen stationären Log-Likelihoodwert zu münden.

- Obwohl es aus numerischer/computationeller Sicht nicht wichtig erscheint, ob eine EM-Parameterfolge konvergiert, war diese Fragestellung damals wenigstens theoretisch interessant, da die zwei fehlerhaften Theoreme von Dempster et al. (1977) über die *Parameter-Konvergenz* des EM-Algorithmus oft zitiert wurden.

In einer Serie von sechs Theoremen zeigte Wu (1983), dass es relativ milde Bedingungen gibt, die einen GEM-Algorithmus dazu zwingen können, (i) dass die Log-Likelihoodwerte seiner Parameterfolge gegen ein *lokales Maximum* der Log-Likelihood konvergiert, und (ii)

dass die Parameterfolge selbst konvergiert. Alle Theoreme machen die folgenden Annahmen, ohne sie jeweils explizit aufzuführen.

Voraussetzungen (Wu (1983)):

- $\Omega \subseteq \mathbb{R}^d$, d.h. der Parameterraum ist eine Teilmenge eines mehrdimensionalen euklidischen Raumes,
- $\{ \Theta \in \Omega \mid L(\Theta) \geq L(\Theta_0) \}$ ist für alle $\Theta_0 \in \Omega$ mit $L(\Theta) > -\infty$ *kompakt* (d.h. *abgeschlossen* und *beschränkt*),
- Die Log-Likelihood $L(.)$ ist in Ω stetig und im *Inneren* $\Omega \backslash \partial\Omega$ *des Parameterraums* differenzierbar.
- Die Parameterfolge eines EM-Algorithmus oder eines GEM-Algorithmus besteht aus lauter *inneren Punkten des Parameterraums.* Dies ist z.B. erfüllt, falls für jeden (Start-)Parameter gilt, dass:

$$\{ \Theta \in \Omega \mid L(\Theta) \geq L(\Theta_0) \} \subseteq \Omega \setminus \partial\Omega \; .$$

Die folgende Definition ist zentral, da auf ihr Theorem 1 (Wu 1983) beruht.

Definition: Sei $\mathcal{A} \subseteq \mathbb{R}^d$. Die Abbildung $\mathcal{M} : \mathcal{A} \to \mathcal{P}(\mathcal{A})$ heisst *abgeschlossen* an der Stelle $a \in \mathcal{A}$, falls aus

$$a_i \to a \quad (i \to \infty)$$

und

$$b_i \to b \quad (i \to \infty)$$

mit $a_i \in \mathcal{A}$ und $b_i \in \mathcal{M}(a_i)$ folgt, dass:

$$b \in \mathcal{M}(a) \; .$$

Das folgende Theorem ist die allgemeinste bekannte Aussage für einen EM-Algorithmus oder einen GEM-Algorithmus. Es zeigt, dass Log-Likelihoodwerte der erzeugten Parameterfolge unter milden Voraussetzungen gegen einen stationären Wert der Log-Likelihood konvergieren und unter strengeren Voraussetzungen sogar gegen ein lokales Maximum.

Theorem (Theorem 1 aus Wu (1983)): Sei $\mathcal{S}$ (bzw. $\mathcal{M}ax$) die Menge der stationären Punkte (bzw. lokalen Maxima) der Log-Likelihood, die im Inneren $\Omega \setminus \partial\Omega$ des Parameterraums liegen. Sei

$$\Theta_0,\ \Theta_1,\ \Theta_2,\ \Theta_3,\ \ldots$$

die Parameterfolge eines GEM-Algorithmus, sodass:

$$\Theta_i \in \mathcal{M}(\Theta_{i-1}) \qquad (i = 1, 2, 3, \ldots) \ .$$

Hierbei sei (i) $\mathcal{M}(.)$ eine *abgeschlossene* Abbildung über dem Komplement von $\mathcal{S}$ (bzw. $\mathcal{M}ax$), und (ii)

$$L(\Theta_i) > L(\Theta_{i-1}) \qquad \text{für alle } \Theta_{i-1} \notin \mathcal{S} \text{ (bzw. } \mathcal{M}ax) \ .$$

Dann gilt:

- Jeder *Häufigkeitspunkt* (Grenzwert einer konvergeten Teilfolge) der Parameterfolge

  $$\Theta_0,\ \Theta_1,\ \Theta_2,\ \Theta_3,\ \ldots$$

  ist ein *stationärer Punkt* (bzw. lokales Maximum) der Log-Likelihood.

- Die zu der Parameterfolge gehörige Folge der Log-Likelihoodwerte

  $$L(\Theta_0),\ L(\Theta_1),\ L(\Theta_2),\ L(\Theta_3),\ \ldots$$

  konvergiert monoton wachsend gegen ein:

  $$L^* = L(\Theta^*)$$

  mit $\Theta^* \in \mathcal{S}$ (bzw. $\mathcal{M}ax$).

Beweis: Anwendung eines Theorems von Zangwill (1969), Seite 91.

Für einen EM-Algorithmus (nicht aber für einen GEM-Algorithmus) liegt Konvergenz gegen stationäre Punkte (nicht aber gegen lokale Maxima) schon unter sehr schwachen Bedingungen vor. Um dies zu zeigen, benötigt man den folgenden Schlüsselschritt:

Lemma: Für alle Parameter $\Phi \notin \mathcal{S}$ gilt in einem EM-Algorithmus:

$$L(\mathcal{M}(\Phi)) > L(\Phi) \ .$$

Beweis: Es reicht zu zeigen, dass eine der beiden Eigenschaften in Theorem 1 aus Dempster et al. (1977) verletzt sind. Sei hierzu $\Theta \in \Omega$ zunächst ein beliebiger Parameter. Mit dem Lemma über die Log-Likelihood und die EM-Hilfsfunktion folgt durch Differenzieren, sowie durch Anwendung von Lemma 2 aus Dempster et al. (1977):

$$\nabla_\Theta L(\Theta)|_{\Theta=\Phi} \;=\; \nabla_\Theta Q_\Phi(\Theta)|_{\Theta=\Phi} \;.$$

Für $\Phi \notin \mathcal{S}$ folgt hieraus (und per Definintion von $\mathcal{S}$):

$$\nabla_\Theta Q_\Phi(\Theta)|_{\Theta=\Phi} \neq 0 \;.$$

Ein allgemeiner Satz aus der Differentialrechnung mehrerer Veränderlicher (Erwe 1962) besagt daher, dass Φ kein *lokales Extremum*, d.h. insbesondere kein lokales Maximum, der EM-Hilfsfunktion Q_Φ sein kann. Als *innerer Punkt* des Parameterraums Ω liegt Φ auch nicht auf dem Parameterrand $\partial\Omega$. Da in einem EM-Algorithmus (per Definition der EM-Parameter-Abbildung) genau eine dieser zwei Eigenschaften auf $\mathcal{M}(\Phi)$ zutrifft, folgt:

$$\mathcal{M}(\Phi) \neq \Phi$$

und:

$$Q_\Phi(\mathcal{M}(\Phi)) > Q_\Phi(\Phi) \;.$$

Also ist die Eigenschaft (ii) in Theorem 1 aus Dempster et al. (1977) verletzt. Es folgt die Behauptung. **q.e.d.**

Theorem (Theorem 2 aus Wu (1983)): Die EM-Hilfsfunktion $Q_\Phi(\Theta)$ sei sowohl *stetig in* Θ, als auch *stetig in* Φ. Dann gilt für einen EM-Algorithmus:

- Jeder *Häufigkeitspunkt* der Parameterfolge

  $$\Theta_0,\ \Theta_1,\ \Theta_2,\ \Theta_3,\ \ldots$$

  ist ein *stationärer Punkt* der Log-Likelihood.

- Die zu einer Parameterfolge gehörige Folge der Log-Likelihoodwerte

  $$L(\Theta_0),\ L(\Theta_1),\ L(\Theta_2),\ L(\Theta_3),\ \ldots$$

  konvergiert monoton wachsend gegen einen *stationären Wert der Log-Likelihood*

  $$L^* = L(\Theta^*) \;.$$

Beweis: Während Voraussetzung (i) in Theorem 1 aus Wu (1983) aus der Stetigkeit der EM-Hilfsfunktionen folgt, ist Voraussetzung (ii) für nicht-stationäre Punkte wegen des vorangegangenen Lemmas erfüllt. **q.e.d.**

Leider ist es (ohne zusätzliche Bedingungen zu benutzen) nicht möglich, einen analogen Beweis für lokale Maxima zu führen. Es ist bemerkenswert, dass die Resultate aus Theorem 2 bereits von Baum et al. (1970) für spezielle Modelle vorgestellt wurden. Die Resultate von Theorem 1 wurden hingegen nicht erfasst, da in Baum et al. (1970) schärfer vorausgesetzt wurde, dass $\mathcal{M}(.)$ stetig ist.

Theorem 4 ist ein einfacher Spezialfall des folgenden Theorems und wird deshalb hier nicht vorgestellt. Theorem 5 stellt für einen GEM-Algorithmus wahrscheinlich immer noch die einzige bekannte Aussage zur Parameter-Konvergenz gegen ein lokales Maximum der Log-Likelihooddar.

Theorem (Theorem 5 aus Wu (1983)): Sei

$$\Theta_0,\ \Theta_1,\ \Theta_2,\ \Theta_3,\ \ldots$$

die Parameterfolge eines GEM-Algorithmus, welche die Voraussetzungen (i) und (ii) aus dem Theorem 1 von Wu (1983) erfüllt. Dann gilt

$$\Theta_i \to \Theta^* \qquad (i \to \infty)\ ,$$

wobei

$$L(\Theta_i) \to L^* \qquad (i \to \infty)\ ,$$

und

$$\Theta^* \in \mathcal{S}(L^*) \quad (\text{ bzw. } \mathcal{M}\text{ax}(L^*)\ )\ ,$$

vorausgesetzt, dass

$$||\Theta_i - \Theta_{i-1}|| \to 0 \qquad (i \to \infty)$$

und

$$\mathcal{S}(L^*) = \{\ \Theta \in \mathcal{S} \mid L(\Theta) = L^*\ \}\ ,$$

$$(\text{ bzw. } \quad \mathcal{M}\text{ax}(L^*) = \{\ \Theta \in \mathcal{M}\text{ax} \mid L(\Theta) = L^*\ \}\ )$$

ist eine *diskrete Menge.*

Beweis: Da eine allgemeine Annahme ist, dass $\{\ \Theta \in \Omega \mid L(\Theta) \geq L(\Theta_0)\ \}$ *kompakt* ist, folgt insbesondere, dass die Parameterfolge

$$\Theta_0,\ \Theta_1,\ \Theta_2,\ \Theta_3,\ \ldots$$

beschränkt ist. Gemäss Theorem 28.1 (Ostrowski 1966) bilden die Häufigkeitspunkte einer beschränkten Folge mit

$$||\Theta_i - \Theta_{i-1}|| \to 0 \qquad (i \to \infty)$$

eine kompakte und zusammenhängende Menge. Gemäss Theorem 1 (Wu 1983) liegen diese Häufungspunkte in $\mathcal{S}(L^*)$ (bzw. $\mathcal{M}\mathrm{ax}(L^*)$). Da weiter angenommen wird, dass diese Menge diskret ist, kann es nur einen einzigen Häufungspunkt geben, d.h. die Parameterfolge konvergiert. **q.e.d.**

Analog zu Theorem 2 benötigt man für die Parameter-Konvergenz gegen stationäre Punkte nur relativ milde Voraussetzungen.

Theorem (Theorem 6 aus Wu (1983)): Die EM-Hilfsfunktion $Q_\Phi(\Theta)$ sei *differenzierbar in* Θ, und $\nabla_\Theta\, Q_\Phi(\Theta)$ sei sowohl *stetig in* Θ, als auch *stetig in* Φ. Sei

$$\Theta_0,\ \Theta_1,\ \Theta_2,\ \Theta_3,\ \ldots$$

die Parameterfolge eines GEM-Algorithmus, sodass jeder Parameter Θ_i ein *stationärer Punkt* der EM-Hilfsfunktion $Q_{\Theta_{i-1}}$ ist:

$$\nabla_\Theta Q_{\Theta_{i-1}}(\Theta)\big|_{\Theta=\Theta_i} = 0\ .$$

Dann gibt es einen stationären Punkt Θ^* mit

$$\Theta_i \to \Theta^* \qquad (i \to \infty)\ ,$$

$$L(\Theta_i) \to L^* \qquad (i \to \infty)\ ,$$

vorausgesetzt

$$||\Theta_i - \Theta_{i-1}|| \to 0 \qquad (i \to \infty)$$

und

$$\mathcal{L}(L^*) = \{\ \Theta \in \Omega \mid L(\Theta) = L^*\ \}$$

ist eine *diskrete Menge.*

Beweis: Mit dem Theorem von Ostrowski (1966) kann wie im Beweis zu Theorem 5 (Wu 1983) gezeigt werden, dass:

$$\Theta_i \to \Theta^* \qquad (i \to \infty)\ ,$$

Die Stationarität von Θ^* folgt aus Behauptung (i) des Theorem 4 (Dempster et al. 1977). **q.e.d.**

Neue Ergebnisse zum EM-Algorithmus

Definition (Injektive Parametrisierung der Analysen): Die Parametrisierung $p_\Theta(.|y)$ der Analysen $X(y)$ heisst *injektiv*, falls:

$$p_\Theta(.|y) = p_\Phi(.|y) \text{ für alle } y \in \mathcal{Y} \text{ mit } \tilde{p}(y) > 0 \quad \Rightarrow \quad \Theta = \Phi \ .$$

Korollar: Der mit einem GEM-Algorithmus und einer *injektiven Parametrisierung der Analysen* re-estimierte Parameter $\mathcal{M}(\Theta)$ erfüllt

$$\mathcal{M}(\Theta) = \Theta \ .$$

oder

$$L(\mathcal{M}(\Theta)) > L(\Theta)$$

Beweis: Gemäss Theorem 1 (Dempster et al. 1977) gilt: $L(\mathcal{M}(\Theta)) \geq L(\Theta)$. Für $L(\mathcal{M}(\Theta)) > L(\Theta)$ sind wir fertig. Sei also $L(\mathcal{M}(\Theta)) = L(\Theta)$. Erneut mit Theorem 1 (Dempster et al. 1977) folgt

$$p_{\mathcal{M}(\Theta)}(.|y) = p_\Theta(.|y) \quad \text{für alle } y \in \mathcal{Y} \text{ mit } \tilde{p}(y) > 0 \ .$$

Da angenommen wird, dass eine injektive Parametrisierung der Analysen vorliegt, folgt: $\mathcal{M}(\Theta) = \Theta$. **q.e.d.**

Korollar: Sei eine *injektive Parametrisierung* $p_\Theta(.|y)$ *der Analysen* $X(y)$ gegeben. Dann gilt für alle Parameter $\Theta \neq \Phi \in \Omega$:

$$H_{\text{cross}}(\Theta;\ \Phi) > H(\Phi) \ .$$

D.h. die erwartete Cross-Entropie der Analysen $H_{\text{cross}}(\Theta;\ \Phi)$ besitzt als Funktion von Θ an der Stelle $\Theta = \Phi$ ein *strenges relatives Minimum.*

Beweis: Folgt aus dem Lemma über die erwartete Cross-Entropie und Entropie aller Analysen. **q.e.d.**

Es ist sehr interessant, dass die Injektivität der Parametrisierung der Analysen dasselbe Resultat liefern kann, wie die Eindeutigkeit der Maximum-Likelihood-Estimierung:

Korollar: Der mit einem GEM-Algorithmus und einer *injektiven Parametrisierung der Analysen* re-estimierte Parameter $\mathcal{M}(\hat{\Theta})$ einer Maximum-Likelihood-Estimierung $\hat{\Theta}$ ist *Parameter-invariant*:

$$\mathcal{M}(\hat{\Theta}) = \hat{\Theta} \ .$$

Beweis: Korollar 1 aus Dempster et al. (1977) liefert:

$$p_{\mathcal{M}(\hat{\Theta})}(.|y) = p_{\hat{\Theta}}(.|y) \quad \text{für alle } y \in \mathcal{Y} \text{ mit } \tilde{p}(y) > 0 \ .$$

Da angenommen wird, dass eine injektive Parametrisierung der Analysen vorliegt, folgt die Behauptung. **q.e.d.**

3.7 Stationäre Punkte im EM-Algorithmus

In Dempster et al. (1977) wurden die grundlegenden Eigenschaften des EM-Algorithmus, bzw. des etwas allgemeineren GEM-Algorithmus, vorgestellt (vergleiche Abschnitt 3.6). Ein GEM-Algorithmus generiert (unter sehr schwachen Voraussetzungen) eine Folge von Modellen mit monoton wachsender Log-Likelihood (Theorem 1), deren Grenzwert ein stationärer Punkt der Log-Likelihood ist, falls vorausgesetzt wird, dass der Grenzwert existiert und alle Modelle stationäre Punkte der EM-Hilfsfunktion sind (Theorem 4).

Stationäre Punkte sind Stellen an denen die ersten partiellen Ableitungen einer Funktion Null sind, und an denen daher ein *lokales Maximum*, ein *lokales Minimum*, oder ein *Sattelpunkt* der Funktion vorliegt. Um zu entscheiden, welcher dieser drei Fälle eintritt, wird im allgemeinen am stationären Punkt die Matrix der zweiten partiellen Ableitungen der Funktion berechnet. Ist diese Matrix *positiv definit* hat die Funktion an der fraglichen Stelle ein lokales Minimum, ist sie hingegen *negativ definit*, so liegt ein lokales Maximum vor. Nur wenn die Matrix weder positiv, noch negativ definit ist, kann (muss aber nicht) der stationäre Punkt ein Sattelpunkt sein.

Zusammen mit den Theoremen von Dempster et al. (1977) ist dies die Motivation, um in diesem Abschnitt die ersten und zweiten partiellen Ableitungen der Log-Likelihood und der EM-Hilfsfunktion vorzustellen. Da beide Funktionen logarithmische Terme enthalten, die Funktionen des Parameters $\Theta \in \Omega$ sind, werden sich die folgende Definition und die damit verbundenen Rechenregeln als sehr nützlich erweisen.

Definition (Δ-Operator): Für eine nicht-negative Funktion $f(\Theta)$ von mehrdimensionalen Parametern $\Theta \in \Omega \subseteq \mathbb{R}^d$ wird für eine Koordinate $1 \leq r \leq d$ die folgende Notation benutzt:

$$\Delta_r f(\Theta) = \partial_r \log\ f(\Theta) = \frac{\partial_r f(\Theta)}{f(\Theta)} \ .$$

Lemma (Rechenregeln des Δ-Operators):

(*i*) Für $f(\Phi) > 0$ gilt: $\Delta_r f(\Theta)|_{\Theta=\Phi} = 0 \iff \partial_r f(\Theta)|_{\Theta=\Phi} = 0$,

(*ii*) Für endliche Mengen I gilt: $\Delta_r \prod_{i \in I} f_i(\Theta) = \sum_{i \in I} \Delta_r f_i(\Theta)$.

(*iii*) Für ganze Zahlen $n \in \mathcal{Z}$ gilt: $\Delta_r (f(\Theta))^n = n \cdot \Delta_r f(\Theta)$.

(*iv*) Für Konstanten $c \in \mathbb{R}$ gilt: $\Delta_r \left(c \cdot f(\Theta)\right) = \Delta_r f(\Theta)$.

Beweis: (i) und (iv) sind trivial, (ii) folgt per Induktion aus der Produktregel der Differentialrechnung, und (iii) ist ein Spezialfall von (ii). **q.e.d.**

Stationäre Punkte der Log-Likelihood des EM-Algorithmus und der EM-Hilfsfunktion

Für die ersten partiellen Ableitungen der Log-Likelihood

$$L(\Theta) = \sum_{y \in \mathcal{Y}} \tilde{p}(y) \cdot \log\ p_\Theta(y) = \tilde{p}\,[\ \log\ p_\Theta\ ]$$

folgt unter Anwendung des Δ-Operators:

$$\begin{aligned} \partial_r L(\Theta) &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \cdot \partial_r \log\ p_\Theta(y) \\ &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \cdot \Delta_r p_\Theta(y) \\ &= \tilde{p}\,[\ \Delta_r p_\Theta\ ]\ . \end{aligned}$$

Es konnte erwartet werden, dass sich die ersten partiellen Ableitungen der Log-Likelihood (wie die Log-Likelihood selbst) als ein empirischer Erwartungswert über unvollständige Datentypen ausdrücken lässt. In den folgenden Schritten soll gezeigt werden, dass sie sich sogar als ein *empirischer Erwartungswert über vollständige Datentypen* ausdrücken lässt. Es gilt:

$$\begin{aligned} \Delta_r p_\Theta(y) &= \frac{\partial_r p_\Theta(y)}{p_\Theta(y)} \\ &= \frac{\partial_r \sum_{x \in X(y)} p_\Theta(x)}{p_\Theta(y)} \\ &= \frac{\sum_{x \in X(y)} \partial_r p_\Theta(x)}{p_\Theta(y)} \\ &= \sum_{x \in X(y)} \frac{\partial_r p_\Theta(x)}{p_\Theta(y)} \\ &= \sum_{x \in X(y)} \frac{p_\Theta(x)}{p_\Theta(y)} \cdot \frac{\partial_r p_\Theta(x)}{p_\Theta(x)} \\ &= \sum_{x \in X(y)} p_\Theta(x|y) \cdot \Delta_r p_\Theta(x) \\ &= p_\Theta(.|y)\,[\ \Delta_r p_\Theta\ ]\ . \end{aligned}$$

Damit folgt:

$$\begin{aligned} \partial_r L(\Theta) &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in X(y)} p_\Theta(x|y) \cdot \Delta_r p_\Theta(x) \\ &= \tilde{p}\,[\ p_\Theta(.|.)\,[\ \Delta_r p_\Theta\ ]\ ]\ . \end{aligned}$$

Unter Benutzung der empirischen Wahrscheinlichkeitsverteilung $\tilde{p}_\Phi$ (hier von $\tilde{p}_\Theta$), die in Abschnitt 3.3 eingeführt wurde, erhält man die folgende schöne Formel für die ersten partiellen Ableitungen der Log-Likelihood des EM-Algorithmus:

$$\begin{aligned} \partial_r L(\Theta) &= \sum_{y \in \mathcal{Y}} \sum_{x \in X(y)} \tilde{p}(y) \cdot p_\Theta(x|y) \cdot \Delta_r p_\Theta(x) \\ &= \sum_{x \in \mathcal{X}} \tilde{p}_\Theta(x) \cdot \Delta_r p_\Theta(x) \\ &= \tilde{p}_\Theta [\ \Delta_r p_\Theta\ ] \ . \end{aligned}$$

Dieses Ergebnis überrascht ein wenig, da es ausdrückt, dass sich die ersten partiellen Ableitungen der Log-Likelihood des EM-Algorithmus sich mit (stochastischen Funktionen von) vollständigen Datentypen ausdrücken lassen, was für die Log-Likelihood selbst nicht der Fall ist.

Die EM-Hilfsfunktion Q_Φ wurde in Abschnitt 3.3 mit der Log-Likelihood L_Φ identifiziert. Deshalb werden im folgenden die ersten partiellen Ableitungen beider Funktionen angegeben. Für

$$\begin{aligned} Q_\Phi(\Theta) &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in X(y)} p_\Phi(x|y) \cdot \log\ p_\Theta(x) \\ &= \tilde{p} [\ p_\Phi(.|.) [\ \log\ p_\Theta\ ]\ ] \end{aligned}$$

gilt

$$\begin{aligned} \partial_r Q_\Phi(\Theta) &= \sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in X(y)} p_\Phi(x|y) \cdot \Delta_r p_\Theta(x) \\ &= \tilde{p} [\ p_\Phi(.|.) [\ \Delta_r p_\Theta\ ]\ ] \ , \end{aligned}$$

während die interpretierte EM-Hilfsfunktion

$$\begin{aligned} L_\Phi(\Theta) &= \sum_{x \in \mathcal{X}} \tilde{p}_\Phi(x) \cdot \log\ p_\Theta(x) \\ &= \tilde{p}_\Phi [\ \log\ p_\Theta\ ] \end{aligned}$$

die ersten partiellen Ableitungen

$$\begin{aligned} \partial_r L_\Phi(\Theta) &= \sum_{x \in \mathcal{X}} \tilde{p}_\Phi(x) \cdot \Delta_r p_\Theta(x) \\ &= \tilde{p}_\Phi [\ \Delta_r p_\Theta\ ] \end{aligned}$$

besitzt. Abbildung 3.21 zeigt eine Übersicht über die ersten partiellen Ableitungen im Kontext des EM-Algorithmus. Es werden die Log-Likelihood des EM-Algorithmus (zweite

	Definition	**Erste partielle Ableitungen**	
$L(\Theta)$	$\tilde{p}\,[\,\log\ p_\Theta\,]$	$\tilde{p}\,[\,p_\Theta(.|.)\,[\,\Delta_r p_\Theta\,]\,]$	$\tilde{p}_\Theta\,[\,\Delta_r p_\Theta\,]$
$Q_\Phi(\Theta)$	$\tilde{p}\,[\,p_\Phi(.|.)\,[\,\log\ p_\Theta\,]\,]$	$\tilde{p}\,[\,p_\Phi(.|.)\,[\,\Delta_r p_\Theta\,]\,]$	
$L_\Phi(\Theta)$	$\tilde{p}_\Phi\,[\,\log\ p_\Theta\,]$		$\tilde{p}_\Phi\,[\,\Delta_r p_\Theta\,]$

Abbildung 3.21: Stationäre Punkte im Kontext des EM-Algorithmus

Zeile), die EM-Hilfsfunktion (dritte Zeile) und die interpretierte EM-Hilfsfunktion (vierte Zeile), sowie ihre Definitionen (zweite Spalte) und ihre ersten partiellen Ableitungen gezeigt (in der dritten Spalte, wenn diese auf unvollständigen Datentypen basieren, bzw. in der vierten Spalte, wenn sie lediglich vollständige Datentypen benutzen). Der Vergleich zeigt, dass die ersten partiellen Ableitungen der Log-Likelihood des EM-Algorithmus sowohl mit vollständigen, als auch unter alleiniger Benutzung von unvollständigen Datentypen dargestellt werden kann. Das Letztere ist ein wenig überraschend (zweite Zeile, vierte Spalte). Der Vergleich zeigt ferner, dass die ersten partiellen Ableitungen der Log-Likelihood des EM-Hilfsfunktion eine starke formale Ähnlichkeit mit den ersten partiellen Ableitungen der (interpretierten) EM-Hilfsfunktion haben. Der Unterschied drückt sich formal durch ein einziges Symbol aus: entweder muss ein Erwartungswert mit dem (unbekannten) Parameter Θ (bei der Log-Likelihood) oder mit dem (bekannten) gegenwärigen Parameter Φ berechnet werden. Leider ist dieser kleine Unterschied dafür verantworlich, dass die praktische Bestimmung der stationären Punkte im ersten Fall (Log-Likelihood des EM-Algorithmus) oft sehr schwierig ist, da hier ungleich kompliziertere Gleichungssysteme zu lösen sind, wie im zweiten Fall (EM-Hilfsfunktion und interpretierte EM-Hilfsfunktion). Dieses mathematische Argument für den EM-Algorithmus wurde bereits in Abschnitt 3.6 angesprochen. Abbildung 3.21 zeigt deutlich, dass es einem Anwender des EM-Algorithmus wohl empfohlen werden kann, die stationären Punkte nicht mit der EM-Hilfsfunktion, sondern mit der interpretierten EM-Hilfsfunktion zu bestimmen (vierte Zeile, vierte Spalte). Da dieses Ergebnis neu ist, soll es in einem Lemma formuliert werden.

Lemma: Bei vorgegebenen gegenwärtigen Parameter $\Phi \in \Omega$ berechnen sich die stationären Punkte der EM-Hilfsfunktion unter Benutzung der ersten partiellen Ableitungen der interpretierten EM-Hilfsfunktion ($r = 1 \ldots d$):

$$\partial_r L_\Phi(\Theta) \;=\; \tilde{p}_\Phi\,[\,\Delta_r p_\Theta\,] \;=\; \sum_{x \in \mathcal{X}} \tilde{p}_\Phi(x) \cdot \Delta_r p_\Theta(x)\ .$$

Lokale Maxima der EM-Hilfsfunktion und der Log-Likelihood des EM-Algorithmus

Um zu entscheiden, ob die Log-Likelihood $L(\Theta)$ an einer Stelle Θ^* ein lokales Maximum besitzt, kann Theorem 5 (Wu 1983) angewendet werden. Möglicherweise wird das folgende Theorem, das im wesentlichen auf den Ergebnissen aus Dempster et al. (1977) beruht, einfacher anzuwenden sein, da Eigenschaft (ii) aus Theorem 1 (Wu 1983) in den meisten Instanzen des EM-Algorithmus wohl nur schwer nachweisbar sein wird.

Theorem: Sei Θ^* der Grenzwert einer in einem GEM-Algorithmus durch iterierte Anwendung der Parameter-Abbildung $\mathcal{M}(.)$ gewonnenen (und mit einem beliebigen Startparameter beginnenden) Parameterfolge. Für den jeweils gegenwärtigen Parameter Φ und den daraus re-estimierten Parameter $\hat{\Theta} = \mathcal{M}(\Phi)$ seien die folgenden Voraussetzungen erfüllt:

(i) die interpretierte EM-Hilfsfunktion $L_\Phi(\Theta)$ besitzt an der Stelle $\Theta = \hat{\Theta}$ einen stationären Punkt, d.h. für $r = 1 \ldots d$ gilt:

$$\partial_r L_\Phi(\Theta)|_{\Theta=\hat{\Theta}} \;=\; 0,$$

(ii) alle Eigenwerte der Matrix der zweiten partiellen Ableitungen der interpretierten EM-Hilfsfunktion $L_\Phi(\Theta)$ an der Stelle $\Theta = \hat{\Theta}$

$$\partial^2_{r,s} L_\Phi(\Theta)\Big|_{\Theta=\hat{\Theta}}$$

sind kleiner oder gleich einem (für die ganze Parameterfolge fixierten) $\lambda < 0$,

(iii) die Matrix der zweiten partiellen Ableitungen der erwarteten Cross-Entropie aller Analysen $H_{\text{cross}}(\Theta;\Theta^*)$ verschwindet an der Stelle $\Theta = \Theta^*$

$$\partial^2_{r,s} H_{\text{cross}}(\Theta;\Theta^*)\Big|_{\Theta=\Theta^*} \;=\; 0\ .$$

Dann hat die Log-Likelihood $L(\Theta)$ an der Stelle $\Theta = \Theta^*$ ein lokales Maximum.
Zur Überprüfung der Voraussetzung (iii) kann

$$\begin{aligned}\partial^2_{r,s}\, H_{\text{cross}}(\Theta,\Theta^*)|_{\Theta=\Theta^*} \;=\;& \tilde{p}\,[\, p_{\Theta^*}(.|y)\,[\, \Delta_r p_\Theta|_{\Theta=\Theta^*} \cdot \Delta_s p_\Theta|_{\Theta=\Theta^*}\,]\,] \\ & -\tilde{p}\,[\, p_{\Theta^*}(.|y)\,[\, \Delta_r p_\Theta|_{\Theta=\Theta^*}\,] \cdot p_{\Theta^*}(.|y)\,[\, \Delta_s p_\Theta|_{\Theta=\Theta^*}\,]\,]\ .\end{aligned}$$

benutzt werden.

Beweis: Es gilt $Q_\Phi(\Theta) = L_\Phi(\Theta)$. (i) und Teil (i) aus Theorem 4 (Dempster et al. 1977)

liefern daher, dass an der Stelle $\Theta = \Theta^*$ ein stationärer Punkt der Log-Likelihood $L(\Theta)$ vorliegt. Um zu prüfen, ob an dieser Stelle sogar ein lokales Maximum vorhanden ist, werden auch die zweiten partiellen Ableitungen berechnet. Offensichtlich gilt:

$$\partial^2_{r,s} L(\Theta)\Big|_{\Theta=\hat{\Theta}} \;=\; \partial^2_{r,s} L_\Phi(\Theta)\Big|_{\Theta=\hat{\Theta}} + \partial^2_{r,s}\; H_{\text{cross}}(\Theta,\Phi)|_{\Theta=\hat{\Theta}} \;.$$

Grenzübergang $\Phi \to \Theta^*$ und $\hat{\Theta} \to \Theta^*$ liefert (wegen der Stetigkeit aller beteiligten Funktionen):

$$\partial^2_{r,s} L(\Theta)\Big|_{\Theta=\Theta^*} \;=\; \partial^2_{r,s} L_{\Theta^*}(\Theta)\Big|_{\Theta=\Theta^*} + \partial^2_{r,s}\; H_{\text{cross}}(\Theta,\Theta^*)|_{\Theta=\Theta^*} \;.$$

Mit (ii) und Teil (ii) aus Theorem 4 (Dempster et al. 1977) folgt, dass die Matrix der zweiten partiellen Ableitungen der EM-Hilfsfunktion an der Stelle $\Theta = \Theta^*$

$$\partial^2_{r,s} L_{\Theta^*}(\Theta)\Big|_{\Theta=\Theta^*}$$

negativ definit ist. (iii) liefert daher, dass auch die Matrix der zweiten partiellen Ableitungen der Log-Likelihood an der Stelle $\Theta = \Theta^*$

$$\partial^2_{r,s} L(\Theta)\Big|_{\Theta=\Theta^*}$$

negativ definit ist. Alles zusammenfassend, muss die Log-Likelihood $L(\Theta)$ an der Stelle $\Theta = \Theta^*$ ein lokales Maximum besitzen.

Es ist nun noch die Rechenregel zur Überprüfung der Voraussetzung (iii) zu beweisen. Für den Rest dieses Beweises sei:

$$\Phi = \Theta^* \;.$$

In Lemma 2 (Dempster et al. 1977) wurde bereits gezeigt, dass:

$$\partial^2_{r,s}\; H_{\text{cross}}(\Theta,\Phi)|_{\Theta=\Phi} \;=\; \tilde{p}\,[\; p_\Phi(.|.)\,[\; \Delta_r p_\Theta(.|.)|_{\Theta=\Phi} \cdot \Delta_s p_\Theta(.|.)|_{\Theta=\Phi} \;]\;] \;.$$

Mit den Rechenregeln des Δ-Operators folgt zunächst:

$$\begin{aligned}
\Delta_r p_\Theta(x|y) &= \Delta_r \frac{p_\Theta(x)}{p_\Theta(y)} \\
&= \Delta_r p_\Theta(x) - \Delta_r p_\Theta(y) \\
&= \Delta_r p_\Theta(x) - p_\Theta(.|y)\,[\; \Delta_r p_\Theta \;] \;.
\end{aligned}$$

Somit:

$$\Delta_r p_\Theta(.|y)|_{\Theta=\Phi} \;=\; \Delta_r p_\Theta|_{\Theta=\Phi} - p_\Phi(.|y)\,[\; \Delta_r p_\Theta|_{\Theta=\Phi} \;] \;.$$

Mit den Rechenregeln des Erwartungswertes folgt:

$$\begin{aligned}
& p_\Phi(.|y)\,[\; \Delta_r p_\Theta(.|y)|_{\Theta=\Phi} \cdot \Delta_s p_\Theta(.|y)|_{\Theta=\Phi} \;] \\
&\quad = p_\Phi(.|y)\,[\; \Delta_r p_\Theta|_{\Theta=\Phi} \cdot \Delta_s p_\Theta|_{\Theta=\Phi} \;] \\
&\quad - p_\Phi(.|y)\,[\; \Delta_r p_\Theta|_{\Theta=\Phi} \;] \cdot p_\Phi(.|y)\,[\; \Delta_s p_\Theta|_{\Theta=\Phi} \;] \;.
\end{aligned}$$

D.h.

$$\begin{aligned}\partial^2_{r,s}\, H_{\text{cross}}(\Theta,\Phi)|_{\Theta=\Phi} \;=\;& \tilde{p}\,[\, p_\Phi(.|y)\,[\, \Delta_r p_\Theta|_{\Theta=\Phi} \cdot \Delta_s p_\Theta|_{\Theta=\Phi}\,]\,] \\ & - \tilde{p}\,[\, p_\Phi(.|y)\,[\, \Delta_r p_\Theta|_{\Theta=\Phi}\,] \cdot p_\Phi(.|y)\,[\, \Delta_s p_\Theta|_{\Theta=\Phi}\,]\,] \;.\end{aligned}$$

q.e.d.

Bemerkung:

1. Um wie im Theorem zu erreichen, dass die Matrix der zweiten partiellen Ableitungen der Log-Likelihood

$$\partial^2_{r,s} L(\Theta)\Big|_{\Theta=\Theta^*} \;=\; \partial^2_{r,s} L_{\Theta^*}(\Theta)\Big|_{\Theta=\Theta^*} + \partial^2_{r,s}\, H_{\text{cross}}(\Theta,\Theta^*)|_{\Theta=\Theta^*}$$

negativ definit ist, reicht es offensichtlich aus, zu verlangen, dass die Matrix der zweiten partiellen Ableitungen der erwarteten Cross-Entropie *negativ semidefinit* ist. Aus dem Lemma über die erwartete Cross-Entropie und Entropie aller Analysen ist aber bekannt, dass

$$H_{\text{cross}}(\Theta;\Theta^*)$$

an der Stelle $\Theta = \Theta^*$ ein lokales Minimum besitzt (siehe auch Abbildung 3.17). Aus Satz 63 (Erwe 1962) folgt daher, dass die Matrix der zweiten partiellen Ableitungen

$$\partial^2_{r,s} H_{\text{cross}}(\Theta;\Theta^*)\Big|_{\Theta=\Theta^*}$$

positiv semidefinit ist. Daher ist die scheinbar schwächere Forderung, dass die Matrix der zweiten partiellen Ableitungen der erwarteten Cross-Entropie *negativ semidefinit* ist, äquivalent zu der im Theorem gemachten Voraussetzung (iii). Zusammenfassend besagt sie, dass die erwartete Cross-Entropie aller Analysen an der Stelle $\Theta = \Theta^*$ kein *strenges lokales Maximum* besitzen darf.

2. Die Voraussetzung (iii) des Theorems

$$\partial^2_{r,s} H_{\text{cross}}(\Theta;\Phi)\Big|_{\Theta=\Phi} \;=\; 0$$

besagt, dass die stochastischen Funktionen $\Delta_s p_\Theta|_{\Theta=\Phi}$ und $\Delta_r p_\Theta|_{\Theta=\Phi}$ bezüglich der bedingten Wahrscheinlichkeitsverteilung $p_\Phi(.|y)$ und unter dem empirischen Erwartungswert $\tilde{p}_\Theta\,[\,.\,]$ stochastisch unabhängig sind:

$$\begin{aligned}& \tilde{p}\,[\, p_\Phi(.|y)\,[\, \Delta_r p_\Theta|_{\Theta=\Phi} \cdot \Delta_s p_\Theta|_{\Theta=\Phi}\,]\,] \\ =\;& \tilde{p}\,[\, p_\Phi(.|y)\,[\, \Delta_r p_\Theta|_{\Theta=\Phi}\,] \cdot p_\Phi(.|y)\,[\, \Delta_s p_\Theta|_{\Theta=\Phi}\,]\,] \;.\end{aligned}$$

3.8 EM-Algorithmus für kontextfreie Grammatiken

In diesem Abschnitt wird der EM-Algorithmus auf probabilistische kontextfreie Grammatiken angewendet. Es wird sich zeigen, dass sich die *re-estimierten Grammatikregel--Wahrscheinlichkeiten* berechnen, indem die *erwartete Häufigkeit einer Grammatikregel* durch die *erwartete Häufigkeit ihrer Mutterkategorie* dividiert wird. Der Erwartungswert muss hierbei auf dem *gegenwärtigen Korpus der vollständigen Datentypen* ermittelt werden. Mit Hilfe der in diesem Abschnitt vorgestellten Re-Estimierungsformeln wird in Abschnitt 3.9 gezeigt werden, dass der *Inside-Outside-Algorithmus* (Baker 1979) eine *dynamische Programmiervariante des EM-Algorithmus* ist. Die wichtigste Folgerung aus dieser Beziehung ist, dass der Inside-Outside-Algorithmus die *Konvergenzeigenschaften des EM-Algorithmus für kontextfreie Grammatiken* erbt. Dieser Beweis wurde bisher noch nie geführt, vielleicht weil häufig falsch zitiert wird, dass ein entsprechender Beweis bereits von Baker (1979) vorgenommen wurde[9].

In diesem Abschnitt werden daher neben den Re-Estimierungsformeln auch sehr genau die Konvergenzeigenschaften dieser Instanz des EM-Algorithmus vorgestellt. Es wird sich zeigen, dass (i) die Log-Likelihood während des Trainings monoton wächst, (ii) dieEM-Hilfsfunktion für die re-estimierten Grammatikregel-Wahrscheinlichkeiten ein *lokales Maximum* annehmen wird, wenn vorausgesetzt werden kann, dass alle Mutterkategorien eine *positive Vorkommenserwartung* aufweisen, und (iii) die Folge der re-estimierten Parameter gegen einen *stationären Punkt* der Log-Likelihood konvergiert. Falls vorausgesetzt werden kann, dass alle Mutterkategorien während der gesamten Re-Estimierung eine positive Vorkommenserwartung haben und die *erwartete Cross-Entropie aller Analysen* kontrollierbar ist, so liegt sogar Konvergenz gegen ein *lokales Maximum* der Log-Likelihood vor. Zusammenfassend kann man bemerken, dass die Verhältnisse sehr viel komplexer sind, als in der Literatur gemeinhin angenommen wird.

Zum Abschluss dieses Abschnitts wird gezeigt, dass das in Abschnitt 2.5 eingeführte *Baumbank-Trainingsverfahren* ein Spezialfall des EM-Algorithmus für annotierte Datentypen und probabilistische kontextfreie Grammatiken ist. Hierfür wird der *EM-Algorithmus für annotierte Datentypen* eingeführt und gezeigt, dass dieser Algorithmus den optimalen Parameter stets bereits nach einer einzigen Iteration findet. Angewandt auf eine Baumbank liefert er die bekannten Re-Estimierungsformeln (siehe Abschnitt 2.5), sowie einen sehr kurzen Beweis der ausgezeichneten Eigenschaft, dass die so estimierten Grammatikregel-Wahrscheinlichkeiten die Baumbank-Wahrscheinlichkeit global maximieren.

[9]Im Gegensatz zu Baker (1979) stellt Baum (1972) den *Forward-Backward-Algorithmus* für Hidden-Markov-Modelle mit Konvergenzaussage und -beweis vor.

Probabilistische kontextfreie Grammatiken

In dieser Arbeit wurden probabilistische kontextfreie Grammatiken und ihre Eigenschaften bereits mehrfach behandelt:

In Abschnitt 2.2 wurden *probabilistische kontextfreie Grammatiken* und die zentralen Begriffe *Grammatikregel-Wahrscheinlichkeit*, *Syntaxbaum-Wahrscheinlichkeit* und *Satz-Wahrscheinlichkeit* eingeführt. Es wurde motiviert, dass die Grammatikregel-Wahrscheinlichkeiten eine *Standard-Nebenbedingung* erfüllen sollten, damit Grammatiken *konsistent* sind, und es wurde bewiesen, dass z.B. *probabilistisches mehrdimensionales Clustering* mit einer probabilistischen kontexfreien Grammatik konsistent ist. In Abschnitt 2.5 wurde das *Baumbank-Trainingsverfahren* und seine *Evaluierung* diskutiert. Es wurden *Baumbanken* und *Baumbank-Grammatiken* an einem Beispiel vorgestellt. Ferner wurde ein neuer Beweis vorgeführt, der zeigt, dass die *Baumbank-Wahrscheinlichkeit* von der Baumbank-Grammatik unter Standard-Nebenbedingung maximiert wird. Der Schlüsselschritt dieses neuen Beweises war, dass die *Baumbank-Wahrscheinlichkeit* ein Wahrscheinlichkeits-Produkt gewisser *Teilkorpora* ist, von denen bekannt ist, dass sie durch ihre jeweiligen empirischen Wahrscheinlichkeitsverteilungen maximiert werden. In Abschnitt 2.6 wurden sehr ausführlich zwei symbolische Parsingverfahren (*Parsewald-* und *CKY-Algorithmus*), sowie drei stochastische Parsingverfahren (*Count-*, *Inside-*, und *Viterbi-Algorithmus*) vorgestellt. Für alle Algorithmen wurde sorgfältig herausgestellt, dass sie *dynamische Programmierung* verwenden und gezeigt, dass alle Algorithmen *effizient* sind, nämlich von der *Ordnung* $O(n^3)$, wobei n die Satzlänge ist. In Abschnitt 2.7 wurde der *Semiring-Parsing-Algorithmus* (Goodman 1998) vorgestellt, der alle in Abschnitt 2.6 eingeführten symbolischen und stochastischen Parsingverfahren verallgemeinert. Es wurde gezeigt, dass der Semiring-Parsing-Algorithmus effizient ist, wenn die Semiring-Operationen in konstanter Zeit vorgenommen werden können. In Abschnitt 2.8 wurden die drei wichtigsten Sprachmodelle vorgestellt: *n-gram-Modelle*, sowie *Markov-* und *Hidden Markov-Modelle*. Es wurde bewiesen, dass n-gram-Modelle weniger allgemein als Markov-Modelle, und diese wiederum weniger allgemein als Hidden Markov-Modelle sind. Es wurde gezeigt, dass alle drei Sprachmodelle durch probabilistische kontextfreie Grammatiken modelliert werden können und es wurde darauf hingewiesen, dass der bekannte *Forward-Backward-Algorithmus* (Baum 1972) sich deshalb als ein einfacher Spezialfall innerhalb der EM-Theorie erweisen wird.

Im Abschnitt 3.1 wurde der *Inside-Outside-Algorithmus* vorgestellt, mit dem es möglich ist, eine gegebene kontextfreie Grammatik auf unannotiertem Text (d.h. ohne Benutzung einer Baumbank) zu trainieren. Es wurde gezeigt, dass sich der Inside-Outside-Algorithmus als die iterative Hintereinanderausführung des *Inside-, Outside- und Conditional-Algorithmus* darstellt, wobei in einem einzelnen Schritt die *bedingten Grammatikregel-*

Wahrscheinlichkeiten für alle Sätze eines gegebenen Textkorpus akkumuliert werden und in neue (bezüglich Textkorpus und kontextfreie Grammatik besser angepasste) Grammatikregel-Wahrscheinlichkeiten umgerechnet werden. Der Inside-Outside-Algorithmus wurde erstmals von Baker (1979) vorgestellt. Leider wird oft falsch zitiert, dass sich in dieser Arbeit auch eine Konvergenzaussage für die Log-Likelihoodwerte der trainierten Grammatiken, und ein entsprechender Konvergenzbeweis finden lassen. Dies ist nicht der Fall, obwohl die probabilistischen kontextfreien Grammatiken tatsächlich gewisse Konvergenz-Eigenschaften besitzen, wie am Ende dieses Abschnittes gezeigt wird.

In Abschnitt 2.2 wurden die folgenden Bezeichnungen für die *symbolischen Komponenten einer probabilistischen kontextfreien Grammatik* G eingeführt[10]:

- $A \in G$ bezeichnet eine *Grammatikkategorie* A der Grammatik G,
- $r \in G$ bezeichnet eine *Grammatikregel* r der Grammtik G,
- $\mathrm{lhs}(r)$ bezeichnet die *linke Seite* einer Grammatikregel $r \in G$,
- $r \in G_A$ bezeichnet eine Grammatikregel $r \in G$ mit $\mathrm{lhs}(r) = A$,
- $x \in \mathcal{T}(G)$ bezeichnet einen *Syntaxbaum* x der Grammatik G,
- $r \in x$ bezeichnet eine *Grammatikregel* r aus dem Syntaxbaum $x \in \mathcal{T}(G)$,
- $y \in \mathcal{L}(G)$ bezeichnet einen *Satz* der Grammatik G,
- $x \in \mathcal{T}(y)$ bezeichnet einen *Syntaxbaum* des Satzes $y \in \mathcal{L}(G)$.

Als *stochastische Komponenten der probabilistischen kontextfreien Grammatik* kommen die folgenden Definitionen hinzu:

- $p(r)$ bezeichnet die Wahrscheinlichkeit einer Grammatikregel $r \in G$,
- $f_r(x)$ bezeichnet die Vorkommenshäufigkeit der Grammatikregel $r \in G$ in dem Syntaxbaum $x \in \mathcal{T}(G)$,
- $f_A(x)$ bezeichnet die Vorkommenshäufigkeit der Grammatikkategorie $A \in G$ in dem Syntaxbaum $x \in \mathcal{T}(G)$, genauer[11]

$$f_A(x) = \sum_{r \in G_A} f_r(x) \ ,$$

[10] In Abschnitt 2.2 wurden einige formale Details ausgelassen. Diese können jedem Standardwerk, wie Hopcroft und Ullman (1979) oder Aho und Ullman (1972) entnommen werden.

[11] Dies setzt voraus, dass x ein Syntaxbaum ist, in dem keine Grammatikkategorien, als *Terminalknoten* vorkommen. Offensichtlich ist dies aber für die Syntaxbäume $x \in \mathcal{T}(y)$ eines Satzes $y \in \mathcal{L}(G)$ der Fall.

- die Wahrscheinlichkeit des Syntaxbaums $x \in \mathcal{T}(G)$ ist

$$p(x) = \prod_{r \in x} p(r)^{f_r(x)} \ ,$$

- die Wahrscheinlichkeit des Satzes $y \in \mathcal{L}(G)$ ist

$$p(y) = \sum_{x \in \mathcal{T}(y)} p(x) \ .$$

Das *Standard-Wahrscheinlichkeitsmodell* für probabilistische kontextfreie Grammatiken macht für die *produktiven* und *erreichbaren* Grammatikkategorien $A \in G$ die folgenden Annahmen:

$$\sum_{r \in G_A} p(r) = 1 \ .$$

Mit dem Theorem von Booth und Thompson (1973) kann mit dieser und einer weiteren Voraussetzung (siehe Abschnitt 2.2) gezeigt werden, dass G *konsistent* ist:

$$\sum_{y \in \mathcal{L}(G)} p(y) = 1 \ .$$

Instantiierung des EM-Algorithmus für probabilistische kontextfreie Grammatiken

Mit diesen Vorgaben gelingt es relativ leicht, den EM-Algorithmus auf probabilistische kontextfreie Grammatiken anzuwenden. Die Abbildung 3.22 zeigt die dafür nötige Instantiierung der symbolischen und stochastischen Elemente des EM-Algorithmus. Die Anwendung der in Abschnitt 3.7 vorgestellten Vorgehensweise zur Bestimmung stationärer Punkte der Log-Likelihood und der EM-Hilfsfunktion führt für probabilistische kontextfreie Grammatiken zu sehr einfachen Re-Estimierungsformeln der Grammatikregel-Wahrscheinlichkeiten.

Lemma (Re-Estimierungsformeln): Der EM-Algorithmus führt für probabilistische kontextfreie Grammatiken zu den folgenden besonders einfachen *Re-Estimierungsformeln für Grammatikregel-Wahrscheinlichkeiten*:

$$\hat{\Theta}_r = \frac{\tilde{p}_\Phi [\ f_r\ ]}{\tilde{p}_\Phi [\ f_A\ ]} \qquad (r \in G,\ A = \mathrm{lhs}(r)) \ .$$

Für einen Syntaxbaum $x \in \mathcal{X}$ bezeichnet $\tilde{p}_\Phi(x) = \tilde{p}(y) \cdot p_\Phi(x|y)$ die mit den *gegenwärtigen Grammatikregel-Wahrscheinlichkeiten* $\Phi \in \Omega$ berechnete *empirische Wahrscheinlichkeit*, während $f_A(x) = \sum_{r \in G_A} f_r(x)$ die Vorkommenshäufigkeit einer Grammatikkategorie $A \in G$ in x bezeichnet. Für die *re-estimierten Grammatikregel-Wahrscheinlichkeiten* $\hat{\Theta} \in \Omega$ und die EM-Parameter-Abbildung gilt also $\mathcal{M} :\ \Omega \to \Omega,\ \Phi \mapsto \hat{\Theta}$.

Die Elemente des EM-Algorithmus

$$< \mathcal{Y},\ \tilde{p}(.),\ \mathcal{X},\ X(.),\ \Omega,\ \Omega_0,\ \{p_\Theta(.)|\Theta \in \Omega\},\ \mathcal{M}(.) >$$

werden für eine probabilistische kontextfreie Grammatik G wie folgt instantiiert:

- $\mathcal{Y} = \mathcal{L}(G)$, die Menge der unvollständigen Datentypen besteht aus den Sätzen der Grammatik G,
- $\tilde{p}(.)$, die empirische Wahrscheinlichkeitsverteilung der unvollständigen Datentypen berechnet sich aus den grammatischen Sätzen eines grossen vorgegebenen Trainingskorpus von freiem Text,
- $\mathcal{X} = \mathcal{T}(G)$, die Menge der vollständigen Datentypen besteht aus den Syntaxbäumen der Grammatik G,
- $X(y) = \mathcal{T}(y)$, die symbolische Analyse-Komponente ist ein *Parser*, der zu jedem grammatischen Satz dessen Syntaxbäume liefert,
- $\Omega = \{\ \Theta \in [0,1]^{|G|} \mid g_A(\Theta) = 0\ \}$, ein *Parameterraum mit Nebenbedingungen.* Mit $|G|$ wird die Anzahl der Grammatikregeln $r \in G$ bezeichnet. Die Parametrisierung $\Theta_r = p(r)$ und die Annahme des Standard-Wahrscheinlichkeitsmodells ergeben die *Parameter-Nebenbedingungen*:
$$g_A(\Theta) = 1 - \sum_{r \in G_A} \Theta_r \qquad (A \in G)\ ,$$
- Ω_0, die Startparameter ergeben sich durch eine zufällige Auswahl aus dem eben spezifizierten Parameterraum,
- die parametrisierten Wahrscheinlichkeitsverteilungen der vollständigen Datentypen
$$p_\Theta(x) = \prod_{r \in G} \Theta_r^{f_r(x)}$$
ergeben sich mit der Parametrisierung $\Theta_r = p(r)$ aus dem Standard-Wahrscheinlichkeitsmodell,
- die EM-Parameter-Abbildung $\mathcal{M} :\ \Omega \to \Omega,\ \Phi \mapsto \hat{\Theta}$ maximiert die (interpretierte) EM-Hilfsfunktion:
$$\begin{aligned} \hat{\Theta} &= \text{argmax}_{\Theta \in \Omega}\ L_\Phi(\Theta) \\ &= \text{argmax}_{\Theta \in \Omega}\ \tilde{p}_\Phi[\ \log\ p_\Theta\ ]\ . \end{aligned}$$

Abbildung 3.22: Der EM-Algorithmus für probabilistische kontextfreie Grammatiken

Beweis: Gemäss Abbildung 3.22 gibt es zu jedem $r \in G$ ein $c \in \mathbb{R}$ mit:

$$p_\Theta(x) = c \cdot \Theta_r^{f_r(x)} \ ,$$

wobei c nicht von Θ_r abhängt. Mit den Rechenregeln des Δ-Operators (siehe Abschnitt 3.7) folgt daher:

$$\begin{aligned} \Delta_r p_\Theta(x) &= \Delta_r \left(c \cdot \Theta_r^{f_r(x)} \right) \\ &= \Delta_r \ \Theta_r^{f_r(x)} \\ &= f_r(x) \cdot \Delta_r \ \Theta_r \\ &= \frac{f_r(x)}{\Theta_r} \ . \end{aligned}$$

Gemäss Abschnitt 3.7 berechnen sich die stationären Punkte der interpretierten EM-Hilfsfunktion unter Benutzung von

$$\begin{aligned} \partial_r L_\Phi(\Theta) &= \tilde{p}_\Phi \left[\ \Delta_r p_\Theta \ \right] \\ &= \tilde{p}_\Phi \left[\ \frac{f_r}{\Theta_r} \ \right] \\ &= \frac{1}{\Theta_r} \cdot \tilde{p}_\Phi \left[\ f_r \ \right] \end{aligned}$$

Da die Parameter $\Theta \in [0,1]^{|G|}$ die Parameter-Nebenbedingungen $g_A(\Theta) = 0$ erfüllen sollen, reicht es leider nicht aus, die lokalen Maxima $\hat{\Theta} \in \Omega$ der interpretierten EM-Hilfsfunktion mit dem einfachen

$$\partial_r L_\Phi(\Theta)|_{\Theta=\hat{\Theta}} = 0$$

zu bestimmen. Stattdessen müssen sie mit der tiefen *Multiplikatorregel* von **J.L. Lagrange** berechnet werden (siehe Abschnitt 2.4), die hier besagt, dass es zu jeder Nebenbedingung $g_A(\Theta) = 0$ einen *Multiplikator* $\lambda_A \in \mathbb{R}$ gibt, sodass das Gleichungssystem

$$\partial_r L_\Phi(\Theta)|_{\Theta=\hat{\Theta}} \ + \ \sum_{A \in G} \lambda_A \cdot \partial_r g_A(\Theta)|_{\Theta=\hat{\Theta}} \ = \ 0 \qquad (r \in G)$$

erfüllt ist. Unter Benutzung der bereits berechneten ersten partiellen Ableitungen von $L_\Phi(\Theta)$ und

$$\partial_r \ g_A(\Theta) = \begin{cases} -1 & \text{für } r \in G_A \\ 0 & \text{sonst} \end{cases}$$

kann dieses Gleichungssystem vereinfacht werden:

$$\frac{1}{\hat{\Theta}_r} \cdot \tilde{p}_\Phi \left[\ f_r \ \right] \ - \ \lambda_A \ = \ 0 \qquad (r \in G, \ A = \mathrm{lhs}(r)) \ .$$

Dies liefert die Lösung:

$$\hat{\Theta}_r \;=\; \frac{\tilde{p}_\Phi [\; f_r \;]}{\lambda_A} \qquad (r \in G,\; A = \mathrm{lhs}(r)) \;.$$

Aus der Parameter-Nebenbedingung $g_A(\hat{\Theta}) = 0$ ergibt sich hieraus noch (per Summation über die $r \in G_A$):

$$\lambda_A = \sum_{r \in G_A} \tilde{p}_\Phi [\; f_r \;] = \tilde{p}_\Phi \left[\sum_{r \in G_A} f_r \right] \;.$$

Die letzte Umformung ist eine Rechenregel des Erwartungswerts. Die letzten beiden Gleichungen ergeben zusammen die Behauptung. **q.e.d.**

Eigenschaften des EM-Algorithmus für probabilistische kontextfreie Grammatiken

Das nächste Theorem ist neu und stellt die wichtigsten Eigenschaften des EM-Algorithmus für probabilistische kontextfreie Grammatiken vor.

Theorem: Sei G eine probabilistische kontextfreie Grammatik. Seien $\hat{\Theta} \in \Omega$ die mit dem EM-Algorithmus aus den gegenwärtigen Grammatikregel-Wahrscheinlichkeiten $\Phi \in \Omega$ re-estimierten Grammatikregel-Wahrscheinlichkeiten (siehe Lemma). Dann gelten die folgenden Aussagen:

(i) Die Log-Likelihood $L(\Theta)$ des EM-Algorithmus wächst während der Re-Estimierung monoton:

$$L(\hat{\Theta}) \geq L(\Phi) \ ,$$

(ii) Die interpretierte EM-Hilfsfunktion $L_\Phi(\Theta)$ hat an der Stelle $\Theta = \hat{\Theta}$ ein *lokales Maximum*, wenn für alle Grammatikkategorien $A \in G$ gilt:

$$\tilde{p}_\Phi [\ f_A\ ] > 0 \ ,$$

(iii) Ist Θ^* der Grenzwert einer durch iterative Re-Estimierung erhaltenen Parameterfolge des EM-Algorithmus, so hat die Log-Likelihood an der Stelle Θ^* einen *stationären Wert* (ein lokales Minimum oder Maximum oder einen Sattelpunkt). Existiert zu jeder Grammatikkategorie $A \in G$ eine Konstante $\lambda \in \mathbb{R}$ mit:

$$\tilde{p}_\Phi [\ f_A\ ] \ > \ \lambda \ > \ 0 \qquad \text{(für alle } \Phi \text{ der Parameterfolge)} \ ,$$

so hat die Log-Likelihood $L(\Theta)$ des EM-Algorithmus an der Stelle $\Theta = \Theta^*$ ein *lokales Maximum*, falls $(r, s \in G)$

$$\partial^2_{r,s} H_{\text{cross}}(\Theta;\ \Theta^*)\Big|_{\Theta=\Theta^*} \ = \ 0 \ .$$

Beweis: Der re-estimierte Parameter $\hat{\Theta}$ ist ein stationärer Punkt der interpretierten EM-Hilfsfunktion (siehe Beweis des letzten Lemmas). Somit folgt (i) aus Theorem 1 (Dempster et al. 1977), und (iii) folgt aus (ii) und dem Theorem aus Abschnitt 3.7. Es bleibt (ii) zu zeigen.

Hierzu werden die zweiten partiellen Ableitungen der interpretierten EM-Hilfsfunktion $L_\Phi(\Theta)$ an der Stelle $\Theta = \hat{\Theta}$ bestimmt. Es gilt $(r, s \in G)$:

$$\partial^2_{s,r} L_\Phi(\Theta) \ = \ \tilde{p}_\Phi [\ \partial_s \Delta_r p_\Theta\ ] \ .$$

Es wurde bereits gezeigt, dass:

$$\Delta_r p_\Theta(x) \ = \ \frac{f_r(x)}{\Theta_r} \ .$$

Somit kann die noch wichtig werdende Beobachtung getroffen werden, dass:

$$\partial_s \Delta_r p_\Theta(x) = 0 \text{ für } s \neq r \ .$$

Für $s = r$ folgt:

$$\partial_s \Delta_r p_\Theta(x) = -\frac{1}{\Theta_r^2} \cdot f_r(x) \ .$$

Zusammenfassend:

$$\partial^2_{s,r} L_\Phi(\Theta)\Big|_{\Theta=\hat{\Theta}} = \begin{cases} 0 & \text{falls } s \neq r \\ -\frac{1}{\hat{\Theta}_r^2} \cdot \tilde{p}_\Phi[\ f_r\ ] & \text{falls } s = r \end{cases}$$

Unter Benutzung der Re-Estimierungsformeln folgt ($A = \mathrm{lhs}(r)$):

$$\partial^2_{s,r} L_\Phi(\Theta)\Big|_{\Theta=\hat{\Theta}} = \begin{cases} 0 & \text{falls } s \neq r \\ -\tilde{p}_\Phi[\ f_A\ ] \cdot \frac{\tilde{p}_\Phi[\ f_A\ ]}{\tilde{p}_\Phi[\ f_r\ ]} & \text{falls } s = r \end{cases}$$

Aus $\frac{\tilde{p}_\Phi[\ f_A\ ]}{\tilde{p}_\Phi[\ f_r\ ]} \geq 1$ und der Voraussetztung $\tilde{p}_\Phi[\ f_A\ ] > 0$ folgt, dass die Matrix

$$\partial^2_\Theta L_\Phi(\Theta)\Big|_{\Theta=\hat{\Theta}}$$

nur aus Diagonalelementen besteht, deren Werte alle wohldefiniert und echt kleiner als Null sind. Somit ist diese Matrix *negativ definit*. **q.e.d.**

Der Beweis des Theorems ist relativ einfach. Dies liegt vor allem daran, dass die parametrisierten Wahrscheinlichkeitsverteilungen $p_\Theta(x)$ der vollständigen Datentypen in dieser Instanz des EM-Algorithmus die Eigenschaft besitzen, dass $\partial_s \Delta_r p_\Theta(x)$ Null ist, wenn s und r verschiedene Grammatikregeln sind. Mit dieser Eigenschaft kann die Matrix der zweiten partiellen Ableitungen der EM-Hilfsfunktion als Diagonalmatrix dargestellt werden, sodass leicht entscheidbar ist, ob die interpretierte EM-Hilfsfunktion $L_\Phi(\Theta)$ an der Stelle $\Theta = \hat{\Theta}$ ein *lokales Maximum* besitzt und eine Parameterfolge des EM-Algorithmus gegen ein lokales Maximum der Log-Likelihood konvergiert. Dies könnte ein sehr guter Hinweis sein, in welchen Fällen der EM-Algorithmus prinzipiell erfolgreich bzw. nicht erfolgreich angewendet werden kann. Die folgende Definition benennt daher solche Wahrscheinlichkeitsverteilungen.

Definition (Parameterseparierbarkeit): Die parametrisierte Wahrscheinlichkeitsverteilung $p_\Theta(x)$ der vollständigen Datentypen heisst *parameterseparierbar*, falls

$$\partial_s \Delta_r p_\Theta(x) = 0 \qquad \text{für } s \neq r \ .$$

Bemerkung:

1. Die Bedingung

   $$\tilde{p}_\Phi[\ f_A\ ] > 0$$

   ist sehr schwach. Sie ist erfüllt, wenn es zu jeder Grammatikkategorie $A \in G$ einen Satz $y \in \mathcal{Y}$ im Trainingskorpus gibt ($\tilde{p}(y) > 0$), sodass (i) ein Syntaxbaum $x \in$

$X(y)$ dieses Satzes die Grammatikkategorie enthält und (ii) dieser Syntaxbaum eine positive Modellwahrscheinlichkeit besitzt ($p_\Phi(x) > 0$ bzw. daraus folgend $p_\Phi(x|y) > 0$):

$$\tilde{p}_\Phi [\, f_A \,] \;\geq\; \tilde{p}_\Phi(x) \cdot f_A(x) \;=\; \tilde{p}(y) \cdot p_\Phi(x|y) \cdot f_A(x) \;>\; 0 \;.$$

2. Vermutlich ist auch die Bedingung

$$\partial^2_{r,s} H_{\text{cross}}(\Theta;\; \Theta^*)\Big|_{\Theta=\Theta^*} \;=\; 0$$

nicht allzu stark, wie die Bemerkungen zum Theorem aus Abschnitt 3.7 vermuten lassen. Für das Training von probabilistischen kontexfreien Grammatiken bietet es sich daher an, in jeder Iteration nicht nur die Log-Likelihoodwerte, sondern auch die Werte der zweiten partiellen Ableitungen der erwarteten Cross-Entropie aller Analysen zu überwachen. Die entsprechenden Berechnungsformeln können zu diesem Zweck dem Theorem aus Abschnitt 3.7 entnommen werden.

Das Baumbank-Trainingsverfahren als Spezialfall des EM-Algorithmus für annotierte Datentypen und probabilistische kontextfreie Grammatiken

Das Ziel dieses Abschnitts ist es, zu zeigen, dass das *Baumbank-Trainingsverfahren* ein Spezialfall des *EM-Algorithmus für annotierte Datentypen* und probabilistische kontextfreie Grammatiken ist.

In Abschnitt 2.5 wurde das *Baumbank-Trainingsverfahren* vorgestellt und ein neuer Beweis vorgeführt, der zeigt, dass die *Baumbank-Wahrscheinlichkeit* von der Baumbank-Grammatik unter Standard-Nebenbedingung maximiert wird. Der Schlüsselschritt dieses neuen Beweises war, dass die *Baumbank-Wahrscheinlichkeit* ein Wahrscheinlichkeits-Produkt gewisser *Teilkorpora* ist, von denen bekannt ist, dass sie durch ihre jeweiligen empirischen Wahrscheinlichkeitsverteilungen maximiert werden.

In diesem Abschnitt soll der Spezialfall des *EM-Algorithmus für annotierte Datentypen* und probabilistische kontextfreie Grammatiken vorgestellt werden. Das Lemma über die Re-Estimierungsformeln der Grammatikregel-Wahrscheinlichkeiten wird dann die in Abschnitt 2.5 aus einer Baumbank erhaltenen Grammatikregel-Wahrscheinlichkeiten liefern, während das letzte Theorem die Aussage liefern wird, dass gerade diese Grammatikregel--Wahrscheinlichkeiten die Baumbank-Wahrscheinlichkeit maximieren. Damit werden die Ergebnisse aus Abschnitt 2.5 ein zweites Mal, aber in einem völlig verschiedenen, und vor allem sehr viel allgemeineren Rahmen bewiesen. Dieses Ergebnis zeigt ein erstes Mal, dass die Theoreme rund um den EM-Algorithmus zwar sehr allgemeingültig formuliert sind, aber doch in der Anwendung zu tiefen Aussagen führen, für deren Beweis sonst ein nicht unerheblicher Aufwand nötig wäre.

In der folgenden Definition wird zunächst der EM-Algorithmus für annotierte Datentypen eingeführt.

Definition (EM-Algorithmus für annotierte Datentypen): Ein EM-Algorithmus ist ein *EM-Algorithmus für annotierte Datentypen*, falls:

- Die Menge der unvollständigen Datentypen fällt mit der Menge der vollständigen Datentypen zusammen:
$$\mathcal{Y} = \mathcal{X} \ .$$
- Es ist ein Trainingskorpus von vollständigen Datentypen gegeben, der durch eine empirische Wahrscheinlichkeitsverteilung repräsentiert wird
$$\tilde{p}(x), \ x \in \mathcal{X} \ .$$
- Die annotierten Datentypen $x \in \mathcal{X}$ sind bereits vollständig annotiert, d.h. $X(x) = \{x\}$. *Im eigentlichen Sinn existiert keine symbolische Analyse-Komponente*, bzw. sie ist trivial:
$$X: \ \mathcal{X} \to \mathcal{P}(\mathcal{X}), \quad x \mapsto \{x\} \ .$$

Man spricht dann von *annotierten Datentypen* $x \in \mathcal{X}$ und dem *Trainingskorpus* $\tilde{p}(x)$ *von annotierten Datentypen.*

Jeder linguistische Korpus, dessen Datentoken in irgendeiner Weise systematisch erstellte linguistische Annotationen tragen, wird diese drei Kriterien erfüllen. Insbesondere gilt dies auch für Baumbanken, die daher auch von dem EM-Algorithmus zum Training einer probabilistischen kontextfreien Grammatik benutzt werden können. Jedes Datentoken dieser Baumbank ist eine (manuell erstellte) linguistische Analyse eines Satzes, die keiner weiteren Annotation bedarf, also ein (manuell disambiguierter) kontextfreier Syntaxbaum der vorgegebenen kontextfreien Grammatik.

Das folgende Lemma ist sehr nützlich. Es hat zur Folge, dass in einem *EM-Algorithmus für annotierte Datentypen* die interpretierte EM-Hilfsfunktion $L_\Phi(\Theta)$ (unabhängig vom gegenwärtigen Parameter $\Phi \in \Omega$) mit der Log-Likelihood $L(\Theta)$ übereinstimmt und daher (i) der optimale Parameter bereits nach einer einzigen Re-Estimierung gefunden wird und (ii) dieser die Log-Likelihood des *Korpus der annotierten Datentypen* maximiert.

Lemma: Ein *EM-Algorithmus für annotierte Datentypen* erfüllt (unabhängig vom gegenwärtigen Parameter $\Phi \in \Omega$):

$$\tilde{p}_\Phi(x) = \tilde{p}(x) \qquad (x \in \mathcal{X}) \ .$$

Beweis: Folgt aus $p_\Phi(x|y) = p_\Phi(x|x) = 1$. **q.e.d.**

Angewandt auf probabilistische kontextfreie Grammatiken liefert dieses Lemma das folgende Korollar, welches überraschend einfach mit Hilfe der allgemeinen EM-Theorie bewiesen werden kann und die Resultate aus Abschnitt 2.5 bestätigt.

Korollar: Sei G eine kontextfreie Grammatik und $\mathcal{X}$ eine Baumbank. Dann führt der *EM-Algorithmus für annotierte Datentypen* zu den folgenden Re-Estimierungsformeln ($r \in G$):

$$\hat{\Theta}_r = \frac{f(r)}{\sum_{r \in G_A} f(r)} \qquad (A = \mathrm{lhs}(r)) \ .$$

Hierbei bezeichnet $f(r) = \sum_{x \in \mathcal{X}} f(x) \cdot f_r(x)$ die *Vorkommenshäufigkeit der Grammatikregel r in der Baumbank $\mathcal{X}$*. Ferner gilt, dass an der Stelle $\Theta = \hat{\Theta}$ ein *globales Maximum* der Log-Likelihood $L(\Theta)$ vorhanden ist:

$$L(\hat{\Theta}) \geq L(\Theta) \qquad (\Theta \in \Omega) \ .$$

Beweis: Unabhängig vom Startparameter $\Phi \in \Omega$ liefert das letzte Lemma mit dem Lemma über die Re-Estimierungsformeln für probabilistische kontextfreie Grammatiken ($r \in G,\ A = \mathrm{lhs}(r)$):

$$\hat{\Theta}_r = \frac{\tilde{p}_\Phi[\ f_r\ ]}{\tilde{p}_\Phi[\ f_A\ ]} = \frac{\tilde{p}[\ f_r\ ]}{\tilde{p}[\ f_A\ ]} \ .$$

Bezeichnet $|f|$ die Anzahl aller Syntaxbäume in der Baumbank $\mathcal{X}$, so folgt:

$$|f| \cdot \tilde{p}[\ f_r\ ] \ = \ |f| \cdot \sum_{x \in \mathcal{X}} \tilde{p}(x) \cdot f_r(x) \ = \ \sum_{x \in \mathcal{X}} f(x) \cdot f_r(x) \ = \ f(r) \ .$$

Dies liefert (mit den Rechenregeln des Erwartungswerts):

$$|f| \cdot \tilde{p}[\ f_A\ ] \ = \ |f| \cdot \tilde{p}\left[\ \sum_{r \in G_A} f_r\ \right] \ = \ \sum_{r \in G_A} |f| \cdot \tilde{p}[\ f_r\ ] \ = \ \sum_{r \in G_A} f(r) \ .$$

Alles zusammenfassend erhält man die im Korollar angegebene Re-Estimierungsformel. Es bleibt zu zeigen, dass die Log-Likelihood $L(\Theta)$ an der Stelle $\Theta = \hat{\Theta}$ ein globales Maximum besitzt. Überraschenderweise kann dies hier schon durch die Anwendung des eher simplen Teil (i) des Theorems gezeigt werden, der insbesondere für die Startparameter liefert, dass

$$L(\hat{\Theta}) \geq L(\Phi) \ .$$

Weil die vorher hergeleitete Re-Estimierungsformel aber zeigt, dass $\hat{\Theta}$ von der speziellen Wahl des Startparameters Φ unabhängig ist, folgt die Behauptung. **q.e.d.**

3.9 Der Inside-Outside-Algorithmus als EM-Algorithmus

Der Inside-Outside-Algorithmus wurde in seiner heute bekannten Form erstmals von Lari und Young (1990) in der Zeitschrift *Computer, Speech and Language* vorgestellt, eine Zeitschrift, die sich hauptsächlich mit Modellen und Experimenten für gesprochene Sprache beschäftigt. In der Arbeit von Lari und Young wurde, nach einer Pause von ca. 10 Jahren, ein Algorithmus von Baker (1979), der ebenfalls für die Sprachtechnologie konzipiert wurde, wiederaufgegriffen und anhand der dort vorgeschlagenen Re-Estimierungsformeln ein Trainingsverfahren für probabilistische kontextfreie Grammatiken vorgestellt, welches nicht wie das bei Baker vorgeschlagene Verfahren auf einen einzigen Satz beschränkt ist, sondern auf *beliebig grossen (satzweise segmentierten) Textkorpora* arbeiten kann. Der von Lari und Young vorgestellte Algorithmus dient auch heute noch, erneut 10 Jahre später, den meisten statistischen Parsern (abgesehen von internen, heuristisch benutzten Glättungsverfahren) als Implementierungsvorlage (siehe zum Beispiel Schmid (1999a)). Die Re-Estimierungsformeln (für einen einzigen Satz) finden sich erstmals in der Arbeit von Baker (1979), der dort die Re-Estimierungsformeln des bekannten *Forward-Backward-Algorithmus* (Baum 1972) für Hidden-Markov-Modelle auf intuitive Weise verallgemeinerte. Obwohl der Forward-Backward-Algorithmus abseits der Linguistik entwickelt wurde, ist er vermutlich das am besten bekannte, und am meisten angewandte stochastische Lernverfahren der Linguistik (siehe Abschnitt 2.8). Der Forward-Backward-Algorithmus arbeitet auf einem Trainingskorpus von Wörtern und hat einige gute formale Eigenschaften, zum Beispiel die Korpuswahrscheinlichkeit in jeder Trainingsiteration zu erhöhen (Monotonie der Log-Likelihood). Dass Baker (1979) die Re-Estimierungsformeln nur für Trainingskorpora aus einem einzigen Satz aufstellte, ist vermutlich dadurch zu erklären, dass der Forward-Backward-Algorithmus auf Wörtern arbeitet, und es lediglich Bakers Ziel war, dem Forward-Backward-Algorithmus zu "kontextfreier Mächtigkeit" zu verhelfen. Entgegen der heute weit verbreiteteten Meinung (siehe zum Beispiel Goodman (1998)) finden sich leider weder in Baker (1979), noch in Lari und Young (1990) Beweise für die erwünschten Eigenschaften probabilistischer kontextfreier Grammatiken (Monotonie der Log-Likelihood, Konvergenz gegen loakle Maxima der Log-Likelihood), wie sie Baum (1972) zum Beispiel für Hidden-Markov-Modelle explizit bewies. Vermutlich gerade wegen dieses Beweises (für Hidden-Markov-Modelle) gingen Baker (1979) und auch Lari und Young (1990) wie selbstverständlich davon aus, dass ihre Algorithmen (für kontextfreie Grammatiken) dieselben theoretischen Eigenschaften besitzen, wobei sich insbesondere Lari und Young (1990) durch eine Serie von Experimenten, in denen sie eine Teilsprache der *Palindrome* mit regulären und kontextfreien Grammatiken darstellten, in dieser Ansicht bestätigt fühlen konnten, da die erwünschte Monotonie in jedem ihrer Experimente auftrat.

In diesem Abschnitt sollen daher die formalen Beweise der allseits bekannten Aussagen über den Inside-Outside-Algorithmus, die man entgegegen der in der Linguistik weitverbreiteten Meinung nicht in der Original-Literatur (und nach bestem Wissen des Autors auch nirgendwo sonst) findet, nachgeholt werden. Hierbei wird der Weg beschritten, formal zu beweisen, dass der Inside-Outside-Algorithmus eine dynamische Programmiervariante des EM-Algorithmus für kontextfreie Grammatiken ist. Auch diese Eigenschaft des Inside-Outside-Algorithmus ist in der Linguistik relativ gut bekannt, aber (nach bestem Wissen des Autors) ebenfalls noch nicht formal bewiesen worden[12]. Zudem ist der exakte Beweis sehr viel weniger leicht, als üblicherweise angenommen wird (Manning und Schütze 1999) und erfordert relativ tiefe Kenntnisse über den Inside-Outside-Algorithmus und den EM-Algorithmus.

Zusammenfassend: Das Ziel dieses Abschnitts ist es, den formalen Beweis der folgenden, in der Linguistik gut bekannten Aussagen über den Inside-Outside-Algorithmus nachzuholen:

- Die Log-Likelihood des Inside-Outside-Algorithmus wächst mononton,
- Die Werte der Log-Likelihood des Inside-Outside-Algorithmus konvergieren gegen einen stationären Wert, in einigen Fällen sogar gegen ein lokales Maximum,
- Der Inside-Outside-Algorithmus ist eine dynamische Programmiervariante des EM-Algorithmus für kontextfreie Grammatiken.

Der Inside-Outside-Algorithmus als dynamische Programmiervariante des EM-Algorithmus für probabilistische kontextfreie Grammatiken

Das folgende Theorem und dessen Beweis zeigen alle gewünschten Eigenschaften des Inside-Outside-Algorithmus.

[12] In Lafferty (1993) wird gezeigt, dass sogenannte *algebraische Modelle* normalisierte erwartete Häufigkeiten als Parameter-Re-Estimierungen besitzen, und es wird darauf hingewiesen (aber nicht formal bewiesen), dass die Grammatikregel-Counts des Inside-Outside-Algorithmus erwartete Häufigkeiten des EM-Algorithmus für kontextfreie Grammatiken sind. Zudem wird lediglich mit dem Inside-Outside-Algorithmus in der Version von Baker (1979) gearbeitet. Somit ist Lafferty der Verdienst zuzuschreiben, thematisiert zu haben, ob der Inside-Outside-Algorithmus eine dynamische Programmiervariante des EM-Algorithmus für kontextfreie Grammatiken ist.

Theorem: Sei G eine kontextfreie Grammatik in Chomsky-Normalform. Seien $\hat{p}(r)$ die mit dem Inside-Outside-Algorithmus aus den gegenwärtigen Grammatikregel-Wahrscheinlichkeiten $p(r)$ re-estimierten Grammatikregel-Wahrscheinlichkeiten. Dann gilt:

(i) Die Log-Likelihood $L(.)$ des Trainingskorpus *wächst monoton*:

$$L(\hat{p}) \geq L(p) \ .$$

(ii) Der Grenzwert einer durch iterative Re-Estimierung erhaltenen Parameterfolge konvergiert gegen einen *stationären Wert* der Log-Likelihood (d.h. gegen ein lokales Minimum oder Maximum oder einen Sattelpunkt). Wenn die *erwartete Cross-Entropie aller Analysen*, sowie die *erwarteten Vorkommenshäufigkeiten der Grammatikkategorien* im Trainingskorpus "kontrollierbar" sind, so liegt sogar ein *lokales Maximum* der Log-Likelihood vor.

(iii) *Der Inside-Outside-Algorithmus ist eine dynamische Programmiervariante des EM-Algorithmus für kontextfreie Grammatiken.* D.h. die re-estimierten Grammatikregel-Wahrscheinlichkeiten $\hat{p}(r)$ des Inside-Outside-Algorithmus stimmen mit den re-estimierten Grammatikregel-Wahrscheinlichkeiten überein, die der EM-Algorithmus für kontextfreie Grammatiken liefert, wenn dieser auf die dieselben gegenwärtigen Grammatikregel-Wahrscheinlichkeiten $p(r)$ angewandt wird.

Beweis: (i) und (ii) folgen mit dem Theorem über den EM-Algorithmus für kontextfreie Grammatiken (siehe Abschnitt 3.8) aus (iii). Zu (iii): Seien $\hat{p}_{EM}(r)$ die mit dem EM-Algorithmus für kontextfreie Grammatiken aus den gegenwärtigen Grammatikregel-Wahrscheinlichkeiten $p(r)$ re-estimierten Grammatikregel-Wahrscheinlichkeiten. Dann gilt für jede Grammatikregel $r \in G$ mit der linken Seite $A = \mathrm{lhs}(r)$, gemäss dem Lemma aus Abschnitt 3.8:

$$\hat{p}_{EM}(r) = \frac{\sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in \mathcal{T}(y)} p(x|y) \cdot f_r(x)}{\sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in \mathcal{T}(y)} p(x|y) \cdot f_A(x)} \ .$$

Ist der Trainingskorpus $\tilde{p}(.)$ durch die Folge $< y_1, \ldots, y_N >$ von Sätzen $y_i \in \mathcal{Y}$ gegeben, so gilt $\tilde{p}(y) = N^{-1} \cdot f(y)$ und es folgt (da sich N^{-1} aus Nenner und Zähler herauskürzt):

$$\begin{aligned} \hat{p}_{EM}(r) &= \frac{\sum_{y \in \mathcal{Y}} f(y) \sum_{x \in \mathcal{T}(y)} p(x|y) \cdot f_r(x)}{\sum_{y \in \mathcal{Y}} f(y) \sum_{x \in \mathcal{T}(y)} p(x|y) \cdot f_A(x)} \\ &= \frac{\sum_{y=y_1}^{y_N} \sum_{x \in \mathcal{T}(y)} p(x|y) \cdot f_r(x)}{\sum_{y=y_1}^{y_N} \sum_{x \in \mathcal{T}(y)} p(x|y) \cdot f_A(x)} \ . \end{aligned}$$

Unter Benutzung der Re-Estimierungsformeln von Lari und Young (1990), die in Ab-

schnitt 3.1 vorgestellt wurden, folgt daher wie gewünscht

$$\hat{p}_{EM}(r) \;=\; \hat{p}(r)\;,$$

vorausgesetzt, dass für jeden Satz $y \in \{y_1 \ldots y_N\}$, sowie für jede Grammatikregel $r \in G$ und jede Grammatikkategorie $A \in G$ gezeigt werden kann, dass:

$$\begin{aligned} C_y(r) &= \sum_{x \in \mathcal{T}(y)} p(x|y) \cdot f_r(x)\;, \\ C_y(A) &= \sum_{x \in \mathcal{T}(y)} p(x|y) \cdot f_A(x)\;. \end{aligned}$$

Dies ist das Ziel des restlichen Beweises, der in zwei Lemmas aufgeteilt wird.

Das erste Lemma geht möglicherweise auf Charniak zurück, der schon bald nach der Arbeit von Lari und Young (1990) entsprechende Formeln in seinem Lehrbuch (Charniak 1993) zur Vorstellung des Inside-Outside-Algorithmus benutze. Das Lemma besagt, dass sich die *Grammatikkategorie-Counts* als Summe der *Grammatikregel-Counts* berechnen, wenn über die Grammatikregeln mit derselben linken Seite summiert wird. Ein expliziter Beweis dieser Aussage wurde bei Charniak leider nicht angegeben (ebensowenig wie eine Referenz auf die Arbeit von Lari und Young (1990)).

Lemma: Für jeden Satz y und jede Grammatikkategorie A gilt (mit den Definitionen aus Abschnitt 3.1):

$$C_y(A) \;=\; \sum_{r \in G,\ \mathrm{lhs}(r)=A} C_y(r)\;.$$

Beweis: In einer kontextfreien Grammatik in Chomsky-Normalform gilt für einen Satz y (der Länge n):

$$\begin{aligned} \sum_{\mathrm{lhs}(r)=A} C_y(r) &= \sum_a C_y(A \to a) \;+\; \sum_{B,C \in G} C_y(A \to B\ C) \\ &:= \sum_a \frac{1}{P} \sum_{1 \le t \le n,\ w_t = a} e(t,t,A)\ f(t,t,A) \\ &\quad + \sum_{B,C \in G} \frac{1}{P} \sum_{s=1}^{n-1} \sum_{t=s+1}^{n} \sum_{r=s}^{t-1} p(A \to BC) e(s,r,B) e(r+1,t,C) f(s,t,A) \\ &= \frac{1}{P} \left(\sum_{1 \le t \le n} e(t,t,A)\ f(t,t,A) \right. \\ &\quad \left. + \sum_{s=1}^{n-1} \sum_{t=s+1}^{n} f(s,t,A) \sum_{B,C \in G} \sum_{r=s}^{t-1} p(A \to BC) e(s,r,B) e(r+1,t,C) \right) \\ &= \frac{1}{P} \left(\sum_{1 \le t \le n} e(t,t,A)\ f(t,t,A) \;+\; \sum_{s=1}^{n-1} \sum_{t=s+1}^{n} f(s,t,A)\ e(s,t,A) \right) \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{P} \sum_{1 \le s \le t \le n} e(s,t,A)\ f(s,t,A) \\
&=: C_y(A)\ .
\end{aligned}
$$

Hierbei wurde in der vierten Gleichung die Rekursionsformel der Inside-Wahrscheinlichkeiten benutzt (siehe Abschnitt 3.1). **q.e.d.**

Dieses Lemma kann benutzt werden, um zu zeigen, dass aus den gesuchten Identitäten für die Grammatikregeln $r \in G$ (durch Summation über alle Regeln mit derselben linken Seite A) die gesuchten Identitäten für die Grammatikkategorien $A \in G$ folgen, da einerseits

$$C_y(A) \ = \ \sum_{A \to \alpha} C_y(A \to \alpha)\ ,$$

und andererseits per Definition

$$f_A(x) \ = \ \sum_{A \to \alpha} f_{A \to \alpha}(x)\ .$$

Der Beweis des Theorems ist daher abgeschlossen, sobald das folgende zentrale Lemma bewiesen ist. Es besagt, dass die *Grammatikregel-Counts* aus dem Inside-Outside-Algorithmus von Lari und Young (1990) nicht nur per Intuition, sondern auch formal mit den *erwarteten Grammatikregel-Vorkommenshäufigkeiten* aus dem EM-Algorithmus identifiziert werden können.

Lemma: Für jeden Satz y und jede Grammatikregel $r \in G$ gilt:

$$C_y(r) \ = \ \sum_{x \in \mathcal{T}(y)} p(x|y) \cdot f_r(x) \ = \ p(.|y)\,[\ f_r\ ]\ .$$

Beweis: Die zweite Gleichung ist die Definition des Erwartungswertes. Da eine kontextfreie Grammatik in Chomsky-Normalform vorliegt, erfolgt der Beweis dieses Lemmas mit Fallunterscheidung.

---

1. Fall: Es liegt eine Grammatikregel der Gestalt $A \to B\ C$ vor:

Für einen gegebenen Satz $y = w_1 \ldots w_n$ und $1 \le s \le r < t \le n$ sei:

$$
\begin{aligned}
X_{(s,t,A)(s,r,B)(r+1,t,C)} \ := \ S \ &\Rightarrow^* \ w_1 \ldots w_{s-1}\ A\ w_{t+1} \ldots w_n \\
&\Rightarrow \ w_1 \ldots w_{s-1}\ B\ C\ w_{t+1} \ldots w_n \\
&\Rightarrow^* \ w_1 \ldots w_r\ C\ w_{t+1} \ldots w_n \\
&\Rightarrow^* \ w_1 \ldots w_n\ .
\end{aligned}
$$

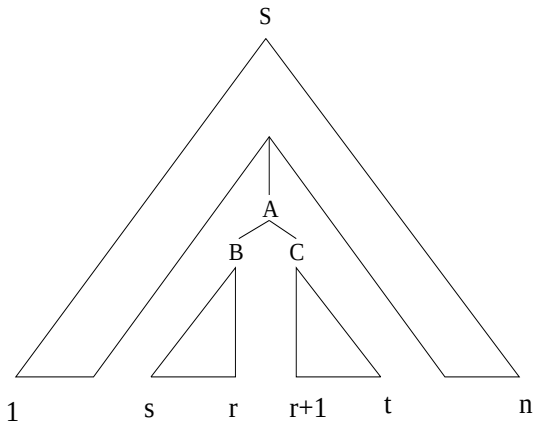


Abbildung 3.23: Dynamische Programmiervariante des EM-Algorithmus für probabilistische kontextfreie Grammatiken.

In Abbildung 3.23 wird dieser durch die drei *Spannen* (s,r,B), $(r+1,t,C)$, und (s,t,A) definierte Parsewald gezeigt. Sei

$$f_{(s,t,A)(s,r,B)(r+1,t,C)}(x) \;:=\; \begin{cases} 1 & \text{falls } x \in X_{(s,t,A)(s,r,B)(r+1,t,C)} \\ 0 & \text{sonst} \end{cases}$$

seine *charakteristische Funktion* auf der Menge $\mathcal{T}(y)$ aller Syntaxbäume des Satzes y.

Die Vorkommenshäufigkeit $f_{A\to BC}(x)$ der Regel $A \to B\ C$ im Syntaxbaum $x \in \mathcal{T}(y)$ berechnet sich damit wie folgt:

$$f_{A\to BC}(x) \;=\; \sum_{1\le s\le r<t\le n} f_{(s,t,A)(s,r,B)(r+1,t,C)}(x)\ .$$

Mit den Rechenregeln des Erwartungswerts folgt daher:

$$\begin{aligned}
p(.|y)\,[\ f_{A\to BC}\ ] &= p(.|y)\left[\ \sum_{1\le s\le r<t\le n} f_{(s,t,A)(s,r,B)(r+1,t,C)}\ \right] \\
&= \sum_{1\le s\le r<t\le n} p(.|y)\left[\ f_{(s,t,A)(s,r,B)(r+1,t,C)}\ \right] \\
&= \sum_{1\le s\le r<t\le n}\ \sum_{x\in\mathcal{T}(y)} p(x|y)\cdot f_{(s,t,A)(s,r,B)(r+1,t,C)} \\
&= \frac{1}{p(y)} \sum_{1\le s\le r<t\le n}\ \sum_{x\in\mathcal{T}(y)} p(x)\cdot f_{(s,t,A)(s,r,B)(r+1,t,C)} \\
&= \frac{1}{p(y)} \sum_{1\le s\le r<t\le n}\ \sum_{x\in X_{(s,t,A)(s,r,B)(r+1,t,C)}} p(x) \\
&= \frac{1}{p(y)} \sum_{1\le s\le r<t\le n} p(X_{(s,t,A)(s,r,B)(r+1,t,C)}) \\
&= \frac{1}{P} \sum_{1\le s\le r<t\le n} f(s,t,A)\cdot p(A\to BC)\cdot e(s,r,B)\cdot e(r+1,t,C) \\
&= C_y(A\to B\ C)\ .
\end{aligned}$$

2. Fall: Es liegt eine Grammatikregel der Gestalt $A \to a$ vor:

Folgt analog (mit den *Spannen* (s, s, A) und (s, s, a)). **q.e.d.**

Kapitel 4

EM-basierte Klassifikationsverfahren

In diesem Kapitel wird ein *probabilistisches Klassifikationsverfahren*[1] für *mehrdimensionale linguistische Datentypen* vorgestellt. Im Gegensatz zu sogenannten "harten Klassifikationsverfahren", die jeden Datentyp in genau eine Klasse stecken, kann in einem probabilistischen Klassifikationsverfahren jeder Datentyp prinzipiell zu mehreren Klassen gehören, da die Klassenzugehörigkeit über gewisse Wahrscheinlichkeitsparameter geregelt wird. Der Begriff *mehrdimensionaler Datentyp* wird im allgemeinen in einer *Domäne* mit zwei oder mehreren endlichen Objektmengen benutzt, in welcher *Vektoren* beobachtet werden, und jedes der Vektorelemente aus einer der Objektmengen stammt. Mehrdimensionale Datentypen kommen in vielen Anwendungen der maschinellen Sprachverarbeitung und in allen Bereichen, von der Phonologie bis zur Pragmatik, vor. Im sogenannten *EM-basierten Klassifikationsverfahren für mehrdimensionale Datentypen* werden *Klassen* als *versteckte Datentypen* angesehen, die aus einem Korpus von *unvollständigen Datentypen* (d.h. von Daten ohne Klassen-Annotationen) unter Benutzung des EM-Algorithmus gelernt werden sollen. Die Hauptaufgabe des EM-basierten Klassifikationsverfahrens für mehrdimensionale Datentypen ist daher

[1] An dieser Stelle muss darauf hingewiesen werden, dass in der vorliegenden Arbeit der deutsche Ausdruck "Klassifikationsverfahren" durchgehend als Übersetzung des englischen Ausdrucks "clustering method" benutzt wird. Dies ist leider nicht ganz unproblematisch, da diese Übersetzung auch mit der Übersetzung des englischen Ausdrucks "classification method" zusammenfällt. Nun ist es aber so, dass man in der englischen Literatur mit "clustering method" ein Verfahren bezeichnet, dass ähnliche Daten zu einer Gruppe zusammenfasst. Hingegen wird eine "classification method" angewandt, wenn man ein einzelnes Datum zu einer Datengruppe zuordnen möchte. Beide Methoden sind zwar eng miteinander verwandt, aber trotzdem die Lösungsverfahren zu zwei verschiedenen Aufgaben.

- die *automatische Enthüllung der verborgenen Klassenstruktur* der Datentypen in einem gegebenen Korpus.

Die Klassifizierung der Datentypen wird durch die Annahme einer sogenannten *Unabhängigkeitsannahme* möglich gemacht. Als Schlüsselidee wird hierbei angenommen, dass innerhalb einer jeden Klasse, alle Objekte miteinander kombinieren können, um Datentypen zu bilden, die genau für diese Klasse charakteristisch sind. Da die Unabhängigkeitsannahme in der Regel nicht für den gesamten Korpus angemessen sein kann, hat sie zur Folge, dass der Korpus in Teile zerbrechen muss, welche die Klassen des Modells bilden werden. Im allgemeinen muss die Anzahl der Klassen experimentell bestimmt werden, um zu erreichen, dass die Unabhängigkeitsannahme optimal getroffen wird. Eine zweite, mit der ersten Aufgabe eng verbundene Aufgabe des EM-basierten Klassifikationsverfahrens ist

- die *Induktion einer glatten Wahrscheinlichkeitsverteilung* über den gegebenen Datentypen.

Für einen gegebenen Korpus, könnten *unbekannte Daten* als diejenigen Datentypen definiert werden, die nicht in diesem Korpus vorkommen, aber in irgendeinem virtuellen Korpus, der für irgendeine virtuelle Anwendung sinnvoll erscheint. Mit dieser Definition werden die realen Daten dieser Welt in zwei Teile zerlegt: (i) die riesige Menge unbekannter Daten und (ii) die winzige Menge der in realen Korpora vorliegenden Daten. Aus diesem Grund ist die wichtigste Eigenschaft aller Wahrscheinlichkeitsmodelle ihre Fähigkeit mit unbekannten Daten umzugehen, d.h. unbekannten (aber sinnvollen) Datentypen eine positive Wahrscheinlichkeit zuzuweisen. Es weist vieles darauf hin, dass EM-basierte Klassifikationsmodelle diese Eigenschaft in besonders vorteilhafter Weise realisieren werden. Zum einen berechnet sich innerhalb dieser Modelle die Wahrscheinlichkeit eines Datentyps als Summe der Wahrscheinlichkeiten dieses Datentyps gegeben seiner Klassen. Ein Datentyp muss also lediglich innerhalb einer einzigen Klasse eine positive Wahrscheinlichkeit besitzen, um eine positive Modell-Wahrscheinlichkeit aufzuweisen, sodass die EM-basierten Klassifikationsmodelle voraussichtlich glatte Wahrscheinlichkeitsverteilungen sein werden. Zum anderen weisen die EM-basierten Klassifikationsmodelle eine *Invarianzeigenschaft* auf, die besagt, dass sich ihre *Marginal-Wahrscheinlichkeitsverteilungen* während der iterativen Bestimmung der Wahrscheinlichkeitsparameter nicht ändern wird. Es kann also angenommen werden, dass die EM-basierten Klassifikationsmodelle nicht nur glatte Wahrscheinlichkeitsverteilungen der empirischen Wahrscheinlichkeitsverteilung sind, sondern zusätzlich die charakteristischen Eigenschaften des gegebenen Korpus behalten werden. Ein weiterer Schwerpunkt dieses Kapitels ist, zu zeigen, dass die EM-basierten Klas-

sifikationsmodelle in vielen Anwendungen der maschinellen Sprachverarbeitung, von der Phonologie bis zur Pragmatik, mit Erfolg eingesetzt werden können. In diesem Kapitel werden vier Anwendungen für EM-basierte Klassifikationsmodelle vorgestellt:

- Aufbau semantisch annotierter Lexika,
- Identifikation von Verb-Nomen-Kollokationen,
- Lexikalische Disambiguierung (in der maschinellen Übersetzung),
- Induktion probabilistischer Silbenmodelle

Die in diesem Kapitel vorgestellte Arbeit ist zum grössten Teil gemeinsame Arbeit. In Rooth et al. (1998) wird gezeigt, dass das in Rooth (1995) vorgestellte probabilistische Klassifikationsverfahren für zwei-dimensionale Verb-Nomen-Paare tatsächlich eine Instanz des EM-Algorithmus ist[2]. In seiner allgemeinsten Form wurde das EM-basierte Klassifikationsverfahren in Müller et al. (2000a) vorgestellt und erfolgreich zum Aufbau drei- und fünfdimensionaler Silbenmodelle benutzt. Der Lexikonaufbau wird in Rooth et al. (1999) und Prescher und Heid (2000) und die Disambiguierungsmethode in Prescher et al. (2000) vorgestellt. Einige neue theoretische und praktische Eigenschaften der EM-basierten Klassifikationsmodelle werden in Prescher (2001d) vorgestellt. Für Forschungszwecke steht eine Implementierung des EM-basierten Klassifikationsverfahrens auf dem Web zur freien Verügung (Prescher 2001a). Dieses Kapitel gliedert sich wie folgt:

In Abschnitt 4.1 werden die theoretischen Grundlagen des EM-basierten Klassifikationsverfahrens für mehrdimensionale Datentypen vorgestellt. Hierzu werden die Begriffe *mehrdimensionaler Datentyp*, *Klasse*, *Marginalverteilung*, *Marginalhäufigkeit*, *Wahrscheinlichkeit eines mehrdimensionalen Datentyps*, *Unabhängigkeitsannahme*, *klassenbasierte Häufigkeit*, *klassenbasierte Marginalhäufigkeit*, *Hyperebene*, und *absolute klassenbasierte Häufigkeit* vorgestellt und motiviert. Damit können die Re-Estimierungsformeln des EM-basierten Klassifikationsverfahrens angegeben werden. Abschliessend wird gezeigt, dass das EM-basierte Klassifikationsverfahren tatsächlich eine Instanz des EM-Algorithmus ist.

In Abschnitt 4.2 werden die formalen Eigenschaften des EM-basierten Klassifikationsverfahrens mit Blick auf drei Ziele untersucht: (i) es werden die *Konvergenzeigenschaften* des EM-basierten Klassifikationsverfahrens betrachtet, wobei unter anderem der Beweis für die in Abschnitt 4.1 angegebenen *Re-Estimierungsformeln* nachgeholt wird, (ii) die *Glättungseigenschaften* der mit dem EM-basierten Klassifikationsverfahren trainierten Modelle werden theoretisch und experimentell überprüft und (iii) es wird eine *Pruning-Technik* vorge-

[2]Pereira et al. (1993) und Hofmann und Puzicha (1998) benutzen dasselbe zweidimensionale Modell, aber andere statistische Inferenzmethoden.

stellt, die dazu führt, dass die mit dem EM-basierten Klassifikationsverfahren trainierten Modelle nur wenig mehr Speicherplatz benötigen, wie die originalen Korpora auf denen sie trainiert wurden.

In Abschnitt 4.3 wird eine zweimalige Anwendung des EM-Algorithmus präsentiert: (i) eine Methode zur Konstruktion von Klassifikationsmodellen für Verb-Nomen-Paare, sowie (ii) eine Methode zur maschinellen *Induktion von semantisch annotierten Subkategorisierungsrahmen*. Die Algorithmen und ihre Implementierungen sind effizient genug, um einen Korpus von ca. 100 Millionen Wörtern in ein Lexikon zu übertragen. Die Methode ist für jede natürliche Sprache einsetzbar, für die es *Texte genügender Grösse*, sowie ein *maschinelles Morphologiesystem* und einen *robusten Parser* gibt, der fähig ist, Subkategorisierungsrahmen mit lexikalischen Köpfen zu extrahieren.

Ziel des Abschnitts 4.4 ist, ein maschinelles Verfahren zur *Identifikation von Verb-Nomen-Kollokationen in Textkorpora* vorzustellen, dass erneut als Schlüsselschritt ein EM-basiertes Klassifikationsmodell benutzt. Da sich Kollokationen gerade dadurch auszeichen, dass (i) die beteiligten Lexeme nur selten auch mit anderen Partnern kombinieren und (ii) die Semantik der Kollokation sich nicht aus der Semantik der beteiligten Lexeme erschliessen lässt, kann das hier vorgestellte, neue Verfahren als Ergänzung des in Abschnitt 4.3 vorgestellten Verfahrens zum Aufbau eines semantisch annotierten Wörterbuchs aufgefasst werden.

Das Ziel des Abschnitts 4.5 ist, zu zeigen, dass durch ein *einfaches Nachschlagen in einem semantisch annotierten Lexikon* (aus Abschnitt 4.3) sehr gute Disambiguierungsresultate erzielbar sind. Jeder Eintrag eines solchen Lexikons besteht im wesentlichen aus einer Wahrscheinlichkeitsverteilung über semantische Klassen, mit denen die Subjekt- und Objektpositionen intransitiver und transitiver Verben annotiert werden. Die Disambiguierung möglicher Zielworte kann mit diesem Lexikon auf eine sehr naheliegende Weise vorgenommen werden, indem im Lexikon nach demjenigen Nomen (unter den möglichen Zielworten) gesucht wird, dessen Semantik am besten mit der Semantik des Zielverbs übereinstimmt. Diese einfache Methode wird auf einer grossen Anzahl von Übersetzungen eines natürlichen bilingualen Korpus evaluiert. Es wird sich zeigen, dass sie ähnlich gute Ergebnisse liefert, wie die von Dagan und Itai (1994), in der aber wesentlich mehr Selektionsinformationen benutzt werden.

In Abschnitt 4.6 werden *probabilistische Silbenmodelle* für Englisch vorgestellt, die mit dem in Abschnitt 4.1 eingeführten mehrdimensionalen EM-basierten Klassifikationsverfahren auf unannotierten drei- und fünfdimensionalen Daten trainiert wurden. Die erhaltenen Silbenmodelle werden mit einem *Pseudo-Disambiguierungsverfahren* evaluiert. Das Ergebnis ist, dass der *Onset* der variabelste Teil der Silbe ist. Die qualitative linguisti-

sche Evaluierung wird zeigen, dass (ohne dass dies eigentlich vorab zu erwarten war) die Silbenmodelle tatsächliche bedeutungstragende Klassen besitzen.

4.1 Grundlagen des EM-basierten Klassifikationsverfahrens

In der Arbeit von Rooth (1995) (und erneut in Rooth (1998)) wird ein probabilistisches Klassifikationsverfahren für zwei-dimensionale Verb-Nomen-Paare vorgestellt, von dem in Rooth et al. (1998) (und kürzer in Rooth et al. (1999)) nachgewiesen wird, dass es sich um ein EM-basiertes Klassifikationsverfahren für zwei-dimensionale Datentypen handelt. Es ist interessant, dass bereits Pereira et al. (1993) und später auch Hofmann und Puzicha (1998) dasselbe zweidimensionale Modell, aber unterschiedliche statistische Inferenzmethoden benutzen. In seiner allgemeinsten Form wurde das EM-basierte Klassifikationsverfahren erstmals in Müller et al. (2000a) vorgestellt und erfolgreich zum Aufbau drei- und fünfdimensionaler Silbenmodelle benutzt.

In diesem Abschnitt wird in die Grundlagen des EM-basierten Klassifikationsverfahrens für mehrdimensionale Datentypen eingeführt, und hierzu die Begriffe *mehrdimensionaler Datentyp*, *Klasse*, *Marginalverteilung*, *Marginalhäufigkeit*, *Wahrscheinlichkeit eines mehrdimensionalen Datentyps*, *Unabhängigkeitsannahme*, *klassenbasierte Häufigkeit*, *klassenbasierte Marginalhäufigkeit*, *Hyperebene*, und *absolute klassenbasierte Häufigkeit* vorgestellt und motiviert. Damit können die Re-Estimierungsformeln des EM-basierten Klassifikationsverfahrens angegeben werden. In einem abschliessenden Teil wird gezeigt, dass das EM-basierte Klassifikationsverfahren tatsächlich eine Instanz des EM-Algorithmus ist.

Einführung in die grundlegenden Begriffe des EM-basierten Klassifikationsverfahrens für mehrdimensionale linguistische Datentypen

Der Begriff *mehrdimensionaler Datentyp* wird im allgemeinen in einer *Domäne* $\mathcal{Y}$ mit zwei oder mehreren endlichen Objektmengen $\mathcal{Y}_i$ benutzt, in welcher *Vektoren* $y \in \mathcal{Y}$ beobachtet werden, und jedes Element aus einer der Objektmengen stammt:

$$y_i \in \mathcal{Y}_i \ .$$

Mehrdimensionale Datentypen kommen in vielen Anwendungen der maschinellen Sprachverarbeitung und auch in allen ihren Bereichen, von der Phonologie bis zur Semantik, vor. Im sogenannten *EM-basierten Klassifikationsverfahren für mehrdimensionale Datentypen* werden *Klassen* als *versteckte Datentypen* angesehen, die aus einem Korpus von unvollständigen Datentypen unter Benutzung des EM-Algorithmus gelernt werden sollen. Die zwei Hauptaufgaben des EM-basierten Klassifikationsverfahrens für mehrdimensionale Datentypen sind

- die *Induktion einer glatten Wahrscheinlichkeitsverteilung* über den gegebenen Datentypen,

- die *automatische Enthüllung der verborgenen Klassenstruktur* der gegebenen Datentypen.

Das Ziel ist es, die Wahrscheinlichkeitsverteilung $p(y)$ aus einem grossen Korpus von Datentypen abzuleiten. Als Schlüsselidee wird dabei angenommen, dass der Datentyp y durch eine *unbeobachtete Klasse* $c \in C$ konditioniert ist, wobei keiner dieser Klassen eine zusätzliche, vorher festgelegte Interpretation gegeben wird. Die Wahrscheinlichkeit eines d-dimensionalen Datentyps

$$y = (y_1, .., y_d)$$

wird dann unter Berücksichtigung dieser Annahme wie folgt definiert:

$$\begin{aligned} p(y) &= \sum_{c \in C} p(c, y) \\ &= \sum_{c \in C} p(c) \cdot p(y|c) \\ &= \sum_{c \in C} p(c) \prod_{i=1}^{d} p(y_i|c) \ . \end{aligned}$$

Während in den ersten beiden Gleichungen keine besonderen mathematischen Annahmen gemacht werden, sondern lediglich die Definition der *Marginalverteilung* (1. Gleichung) und die Definition von *Bayes* (2. Gleichung) benutzt werden, ist es sehr wichtig, festzuhalten, dass in der dritten Gleichung eine relativ starke *Unabhängigkeitsannahme* benutzt wird:

$$p(y|c) = \prod_{i=1}^{d} p(y_i|c) \qquad \text{für alle Klassen } c \in C \ .$$

Intuitiv ist klar, dass diese Unabhängigkeitsannahme die Klassifizierung der Datentypen möglich macht. Sie besagt, dass innerhalb einer jeden Klasse, sämtliche Objekte y_i ohne Ausnahme miteinander kombinieren können, um Datentypen $y = (y_1, .., y_d)$ zu bilden, die genau für diese Klasse charakteristisch sind. Da die Unabhängigkeitsannahme in der Regel nicht für den gesamten Korpus angemessen sein kann, hat sie zur Folge, dass der Korpus in Teile zerbrechen muss, welche die Klassen des Modells bilden werden. Im allgemeinen muss die Anzahl $|C|$ der Klassen experimentell bestimmt werden, um zu erreichen, dass die Unabhängigkeitsannahme optimal getroffen wird. Der EM-Algorithmus (Dempster et al. 1977) zielt darauf ab, die Log-Likelihood

$$L = \sum_{y} \tilde{p}(y) \ln p(y)$$

der unvollständigen Datentypen als eine Funktion der empirischen Wahrscheinlichkeitsverteilung $\tilde{p}$ iterativ zu maximieren. Ist $f(y)$ die empirische Häufigkeit eines mehrdimensionalen Datentyps y, so wird

$$f_c(y) = f(y) \cdot p(c|y)$$

die *klassenbasierte Häufigkeit von y annotiert mit Klasse c* genannt. In diesem Zusammenhang mag interessant sein, dass

$$p(c|y) \;=\; \frac{p(c,y)}{p(y)} \;=\; \frac{p(c)\cdot p(y|c)}{p(y)}$$

(wegen $\sum_c p(c|y) = 1$) als die Wahrscheinlichkeit

$$p(y \in c)$$

interpretiert werden kann, dass der Datentyp y in die Klasse c fällt. Die *klassenbasierten Rand-Häufigkeiten*

$$f_c(y_i) = \sum_{y \in \mathcal{H}(y_i)} f_c(y)$$

eines Objekts

$$y_i \in \mathcal{Y}_i$$

können berechnet werden, indem die klassenbasierten Häufigkeiten aller Datentypen aufsummiert werden, die in der $(d-1)$-*dimensionalen Hyperebene*

$$\mathcal{H}(y_i) = \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_{i-1} \times \{y_i\} \times \mathcal{Y}_{i+1} \times \ldots \times \mathcal{Y}_d$$

liegen. Ferner seien

$$|f| = \sum_{y \in \mathcal{Y}} f(y)$$

die *absolute Häufigkeit* des Korpus, und

$$|f_c| = \sum_{y \in \mathcal{Y}} f_c(y)$$

die *absolute klassenbasierte Häufigkeit* des Korpus aller Datentypen, die mit der Klasse c annotiert sind. Mit diesen Bezeichnungen gilt dann:

Lemma (EM-basiertes Klassifikationsverfahren für mehrdimensionale Datentypen): Das EM-basierte Klassifikationsverfahren für mehrdimensionale Datentypen ist eine Instanz des EM-Algorithmus. Für die re-estimierten Parameter $\hat{p}(c),\ \hat{p}(y_i|c)$ gilt $(c \in C,\ y_i \in \mathcal{Y}_i,\ i = 1,..,d)$:

$$\begin{aligned} \hat{p}(c) &= |f|^{-1} \cdot |f_c| \;, \\ \hat{p}(y_i|c) &= |f_c|^{-1} \cdot f_c(y_i) \;. \end{aligned}$$

Die Formeln decken auf, dass der Re-Estimierungsprozess der EM-basierten Klassifizierung mehrdimensionaler Datentypen auf der wiederholten Anwendung zweier Basistechniken beruht: (i) *Normalisierung*, sowie (ii) *Marginalisierung (via den Elementen einer Hyperebene)*.

Beweis: Der Beweis wird nicht hier geführt. Im nächsten Abschnitt wird gezeigt, dass das EM-basierte Klassifikationsverfahren eine Instanz des EM-Algorithmus ist. In Abschnitt 4.2 wird gezeigt, dass sich die im Lemma angegebenen Re-Estimierungsformeln als Spezialfall der Re-Estimierungsformeln gewisser probabilistischer kontextfreier Grammatiken berechnen. **q.e.d.**

Instantiierung des EM-Algorithmus für die Klassifizierung mehrdimensionaler Datentypen

In diesem Abschnitt soll das EM-basierte Klassifikationsverfahren für mehrdimensionale Datentypen explizit als Instanz des EM-Algorithmus angegeben werden. Abbildung 4.1 zeigt die zu diesem Zweck instantiierten symbolischen und stochastischen Elemente des EM-Algorithmus. Hierbei ist interessant, dass die Menge der unvollständigen Datentypen nur in Relation zu einem vorgegebenen Korpus $f(.)$ von mehrdimensionalen Datentypen definiert ist:

$$\mathcal{Y}_i \;=\; \{\; y_i \mid \text{ es gibt ein } y \in \mathcal{H}(y_i) \text{ mit } f(y) \neq 0 \;\}\;.$$

D.h. die Objektmenge $\mathcal{Y}_i$ ist die Menge derjenigen Objekte y_i, die in einem grossen Trainingskorpus von mehrdimensionalen Beobachtungen in deren i-ter Koordinate vorkommen.

Bemerkungen:

(i) Die Anzahl der Klassen $|C|$ wird nicht direkt durch den EM-Algorithmus festgelegt, sondern ist ein externer Parameter, der in der Regel durch eine sogenannte *Pseudo-Disambiguierung* festgelegt wird (siehe Abschnitt 4.3).

(ii) Trotz ihrer trivialen symbolischen Analyse-Komponente steht die probabilistische Analyse-Komponente $p(c, y),\ (c \in C)$, im Zentrum des Interesses, da sie die Annotationswahrscheinlichkeiten $p(c|y)$ und damit indirekt die klassenbasierten Häufigkeiten

$$f_c(y) = f(y) \cdot p(c|y)$$

liefert. Diese werden sehr erfolgreich zum Disambiguieren lexikalischer Mehrdeutigkeiten eingesetzt (siehe Abschnitt 4.5).

Die Elemente des EM-Algorithmus

$$< \mathcal{Y},\ \tilde{p}(.),\ \mathcal{X},\ X(.),\ \Omega,\ \Omega_0,\ \{p_\Theta(.)|\Theta \in \Omega\},\ \mathcal{M}(.) >$$

werden zur Klassifizierung mehrdimensionaler Datentypen wie folgt instantiiert:

- $\mathcal{Y} = \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_d$, die Menge der unvollständigen Datentypen. Die Objektmenge $\mathcal{Y}_i$ ist die Menge derjenigen Objekte, die in einem grossen Trainingskorpus von mehrdimensionalen Beobachtungen in der i-ten Koordinate vorkommen.

- $\tilde{p}(y)$, die empirische Wahrscheinlichkeitsverteilung der unvollständigen Datentypen berechnet sich aus den Häufigkeiten $f(y)$ von mehrdimensionalen Datentypen y in einem grossen Trainingskorpus.

- $\mathcal{X} = C \times \mathcal{Y}$, die Menge der vollständigen Datentypen. Ein vollständiger Datentyp $x = (c, y)$ besteht aus einem mehrdimensionalenen Datentyp y und einer *Klassenannotation* $c \in C$.

- $X(y) = C \times \{y\}$, die symbolische Analyse-Komponente. Sie ist hier ein trivialer *semantischer Tagger*, der jedem mehrdimensionalen Datentyp die Menge aller möglichen Klassenannotationen zuordnet.

- Ω, ein *Parameterraum mit Nebenbedingungen*:

$$\Omega = \left\{ \Theta =< \Theta_c, \Theta_{c,y_i} >_{c \in C, y_i \in \mathcal{Y}_i} \;\middle|\; \sum_{c \in C} \Theta_c = 1,\ \sum_{y_i \in \mathcal{Y}_i} \Theta_{c,y_i} = 1 \right\} .$$

- Ω_0, die Startparameter ergeben sich durch eine zufällige Auswahl aus dem eben spezifizierten Parameterraum.

- Die parametrisierten Wahrscheinlichkeitsverteilungen der vollständigen Datentypen $p_\Theta(x) = p_\Theta(c, y) = \Theta_c \cdot \prod_{i=1}^{d} \Theta_{c,y_i}$ ergeben sich mit der Parametrisierung $\Theta_c = p(c),\ \Theta_{c,y_i} = p(y_i|c)$ aus dem oben angegebenen Wahrscheinlichkeitsmodell.

- Die EM-Parameter-Abbildung $\mathcal{M} : \Omega \to \Omega, \Theta \mapsto \hat{\Theta}$ maximiert die (interpretierte) EM-Hilfsfunktion:

$$\hat{\Theta} = \text{argmax}_{\Theta \in \Omega}\ L_\Phi(\Theta) = \text{argmax}_{\Theta \in \Omega}\ \tilde{p}_\Phi [\ \log\ p_\Theta\ ] .$$

Abbildung 4.1: Der EM-Algorithmus für die Klassifizierung mehrdimensionaler Daten

4.2 Eigenschaften des EM-basierten Klassifikationsverfahrens

In diesem Abschnitt werden die formalen Eigenschaften des EM-basierten Klassifikationsverfahrens mit Blick auf drei Ziele untersucht: (i) es sollen die Konvergenzeigenschaften des EM-basierten Klassifikationsverfahrens betrachtet werden, wobei unter anderem der Beweis für die in Abschnitt 4.1 angegebenen Re-Estimierungsformeln nachgeholt werden muss, (ii) die Glättungseigenschaften der mit dem EM-basierten Klassifikationsverfahren trainierten Modelle sollen theoretisch und experimentell überprüft werden und (iii) es soll eine Pruning-Technik vorgestellt werden, die dazu führt, dass die mit dem EM-basierten Klassifikationsverfahren trainierten Modelle nur wenig mehr Speicherplatz benötigen, wie die originalen Korpora auf denen sie trainiert wurden.

Es wird sich zeigen, dass das EM-basierte Klassifikationsverfahren als eine probabilistische kontextfreie Grammatik modelliert werden kann, deren Konsistenz bereits in Abschnitt 2.2 mit dem Theorem von Booth und Thompson (1973) nachgewiesen wurde. Die allgemeinen Re-Estimierungsformeln für probabilistische kontextfreie Grammatiken liefern für diese spezielle Grammatik gerade die in Abschnitt 4.1 angegebenen Re-Estimierungsformeln des EM-basierten Klassifikationsverfahrens. Die allgemeinen Konvergenzeigenschaften der probabilistischen kontextfreien Grammatiken übertragen sich also auf das EM-basierte Klassifikationsverfahren. Da die erwarteten Häufigkeiten der Grammatikkategorien gerade mit den absoluten klassenbasierten Häufigkeiten übereinstimmt, wird unter gewissen Voraussetzungen sogar gezeigt werden können, d ass das EM-basierte Klassifikationsverfahren die Log-Likelihood eines Korpus mehrdimensionaler Datentypen maximiert.

Eine der wichtigsten Eigenschaften eines Wahrscheinlichkeitsmodells ist die Fähigkeit mit unbekannten Daten umzugehen, d.h. unbekannten (aber sinnvollen) Datentypen eine positive Wahrscheinlichkeit zuzuweisen. Es wird gezeigt, dass die mit dem EM-basierten Klassifikationsverfahren trainierten Modelle eine *Invarianz-Eigenschaft* besitzen, die besagt, dass die Modelle während des gesamten Re-Estimierungsprozess des EM-Algorithmus die Randverteilungen der empirischen Wahrscheinlichkeitsverteilung der mehrdimensionalen Datentypen aufweisen werden. Da der Beweis ganz wesentlich benutzt, dass das EM-basierte Klassifikationsverfahren eine Instanz des EM-Algorithmus ist, kann vermutet werden, dass in den Ansätzen von Pereira et al. (1993) und Hofmann und Puzicha (1998) (die im zwei-dimensionalen Fall dasselbe Modell, aber eine andere Inferenzmethode benutzen) diese Invarianz-Eigenschaft nicht vorhanden ist. Ferner ist interessant, dass in Kim und Choi (2000) experimentell gezeigt wird, dass die mit SVD (Singular Value

Decomposition[3]) hergestellten zwei-dimensionalen Verb-Nomen-Modelle gute Glättungseigenschaften besitzen. Da SVD lediglich eine Dimensions-Reduktion vornimmt, also auch eine gewisse Invarianz-Eigenschaft hat, kann vermutet werden, dass "hinreichend invariante" Modelle gute Glättungseigenschaften besitzen werden. Für die mit dem EM-basierten Klassifikationsverfahren trainierten Modelle wird dies experimentell nachgewiesen werden: die Glättungskraft der trainierten Modelle ist etwa 500- bis 100-mal besser als die der empirischen Wahrscheinlichkeitsverteilung.

Es wird normalerweise der Fall sein, dass mehr Trainingsdaten die mit dem EM-basierten Klassifikationsverfahren trainierten Modelle verbessern werden. Natürlich wird mit der Grösse des Trainingskorpus auch die Grösse der Modelle anwachsen, was im Extremfall zu Modellen führen kann, die zu gross sind, um in der Praxis benutzt werden zu können. Daher wird eine neue mit klassenbasierten Häufigkeiten arbeitende *Pruning-Technik* vorgestellt, die dazu führen wird, dass die mit dem EM-basierten Klassifikationsverfahren trainierten Modelle nur ungefähr doppelt so viel Speicherplatz benötigen, wie die originalen Korpora.

EM-basierte Klassifizierung mit einer probabilistischen kontextfreien Grammatik

In Abschnitt 2.2 wurde die Grammatik G_{cluster} eingeführt und gezeigt, dass diese Grammatik konsistent ist. In diesem Abschnitt soll die Grammatik erneut betrachtet werden, da sie auf sehr einfache Weise benutzt werden kann, um die Re-Estimierungsformeln für die EM-basierte Klassifizierung mehrdimensionaler Daten herzuleiten (siehe Lemma, Abschnitt 4.1). Die Grammatik G_{cluster} (siehe Abbildung 2.6) hat $|C|+|C|\cdot|\mathcal{Y}_1|+\ldots+|C|\cdot|\mathcal{Y}_d|$ Grammatikregeln, sowie $1+|C|\cdot d$ Grammatikkategorien:

$$S,\ Y_i^c,\ (c \in C,\ i = 1 \ldots d),$$

und das folgende Grammatiklexikon:

$$\mathcal{Y}_1 + \ldots + \mathcal{Y}_d\ .$$

Das $'+'$-Zeichen steht für die Vereinigung von disjunkten Mengen, da (durch Benutzung einer Kodierung, welche die Koordinaten mitberücksichtigt) vorausgesetzt werden kann, dass ein Objekt y_i nur in einer ganz bestimmten Koordinate i vorkommen kann. Die Sprache der Grammatik ist:

$$\mathcal{L}(G_{\text{cluster}}) = \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_d = \prod_{i=1}^{d} \mathcal{Y}_i\ .$$

[3]SVD ist ein Matrizen-Standardverfahren, für welches frei erhältliche Implementierungen vorhanden sind (siehe z.B. Press et al. (1992))

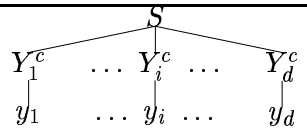
EM-basierte Klassifizierung	Probabilistische kontextfreie Grammatik	per Definition
Wahrscheinlichkeits-Parameter	$p(S \to Y_1^c \dots Y_d^c)$ $p(Y_i^c \to y_i)$	$p(c)$ $p(y_i \mid c)$
Analysen $(c \in C,\ y_i \in \mathcal{Y}_i)$		$(c, y_1 \dots y_d)$
Analyse-Wahrscheinlichkeit	$p(S \to Y_1^c \dots Y_d^c) \cdot \prod_i p(Y_i^c \to y_i)$	$p(c) \cdot \prod_i p(y_i \mid c)$

Abbildung 4.2: EM-basierte Klassifizierung

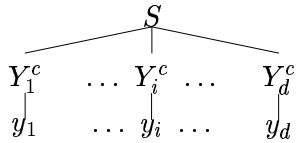
In der Grammatik G_{cluster} hat jeder Satz $y =< y_1 \dots y_d >$ genau $|C|$ nicht-rekursive Syntaxbäume:

$$\mathcal{T}(y) = \left\{ \quad \begin{array}{c} S \\ Y_1^c \quad \dots \quad Y_i^c \quad \dots \quad Y_d^c \\ \downarrow \qquad\quad \downarrow \qquad\quad \downarrow \\ y_1 \quad \dots \quad y_i \quad \dots \quad y_d \end{array} \quad \middle| \quad c \in C \right\}$$

Die Standard-Nebenbedingung hat für G_{cluster} die Gestalt:

$$\sum_{c \in C} p(S \to Y_1^d \dots Y_d^c) = 1, \qquad \sum_{y_i \in \mathcal{Y}_i} p(Y_i^c \to y_i) = 1,\ (c \in C,\ i = 1 \dots d)\ .$$

Der Syntaxbaum



hat die Wahrscheinlichkeit:

$$p(S \to Y_1^c \dots Y_d^c) \cdot \prod_i p(Y_i^c \to y_i)\ .$$

Die Wahrscheinlichkeit eines Satzes $y =< y_1 \dots y_d >$ ist daher:

$$\sum_{c \in C} p(S \to Y_1^c \dots Y_d^c) \cdot \prod_i p(Y_i^c \to y_i)\ .$$

Abbildung 4.2 zeigt einen Vergleich des EM-basierten Klassifizierungsverfahrens mit der Grammatik G_{cluster}. Beide Modelle haben das Ziel, den mehrdimensionalen Datentypen

$$(y_1 \dots y_d) \in \mathcal{Y}_1 \times \dots \times \mathcal{Y}_d$$

eine Wahrscheinlichkeit

$$p(y_1 \dots y_d)$$

zuzuweisen. Obwohl beide Modelle diese Zuweisung mit verschieden Ansätzen vornehmen, macht die Abbildung 4.2 deutlich, dass beide Modelle dieselbe Wahrscheinlichkeitsverteilung über den mehrdimensionalen Datentypen erzeugen werden, wenn die Modellparameter gleiche Werte annehmen: Die vierte Zeile zeigt die Analyse-Wahrscheinlichkeiten (d.h die Wahrscheinlichkeiten der vollständigen Datentypen) der beiden Modelle. In der dritten Zeile werden die Analysen (d.h. die vollständigen Datentypen) und in der zweiten Zeile die Wahrscheinlichkeitsparameter der Analysen (d.h. die Modellparameter) gezeigt. Offensichtlich sind die Analyse-Wahrscheinlichkeiten in beiden Modellen gleich, wenn die Modellparameter übereinstimmen. Da jeder mehrdimensionale Datentyp in beiden Modellen dieselbe Anzahl von Analysen mit denselben Analyse-Wahrscheinlichkeiten besitzt, werden in diesem Fall beide Modelle einem mehrdimensionalen Datentyp dieselbe Wahrscheinlichkeit zuweisen. Als Konsequenz dieser Beobachtung folgt, dass sich die Re-Estimierungsformeln des EM-basierten Klassifizierungsverfahren aus den Re-Estimierungsformeln des EM-Algorithmus ergeben, wenn diese auf die Grammatik G_{cluster} angewendet werden. Dies wird im folgenden getan.

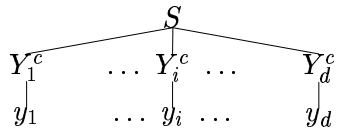
Lemma: Die probabilistische kontextfreie Grammatik G_{cluster} führt bei Anwendung des EM-Algorithmus zu den folgenden Re-Estimierungsformeln ($c \in C,\ y_i \in \mathcal{Y}_i,\ i = 1,..,d$):

$$\begin{aligned} \hat{p}(S \to Y_1^c \ldots Y_d^c) &= |f|^{-1} \cdot |f_c| \ , \\ \hat{p}(Y_i^c \to y_i) &= |f_c|^{-1} \cdot f_c(y_i) \ . \end{aligned}$$

Die klassenbasierten Häufigkeiten werden analog zu Abschnitt 4.1 definiert:

$$f_c(y) := f(y) \cdot p(x_{c,y}|y) \ .$$

Hierbei ist $x_{c,y}$ der Syntaxbaum:



Beweis: Es werden die Re-Estimierungsformeln des EM-Algorithmus für probabilistische kontextfreie Grammatiken angewandt (Theorem, Abschnitt 3.8). Statt der Symbole p_Φ und p_Θ werden die Symbole p und $\hat{p}$ verwendet, und statt der empirischen Wahrscheinlichkeitsverteilung $\tilde{p}$ wird die Häufigkeit $f = |f| \cdot \tilde{p}$ verwendet (da sich die absoluteHäufigkeit f aus Zähler und Nenner herauskürzt):

$$\hat{p}(r) = \frac{\sum_{y \in \mathcal{Y}} f(y) \sum_{x \in \mathcal{T}(y)} p(x|y) \cdot f_r(x)}{\sum_{y \in \mathcal{Y}} f(y) \sum_{x \in \mathcal{T}(y)} p(x|y) \cdot f_A(x)} \qquad (r \in G_{\text{cluster}},\ A = \text{lhs}(r)) \ .$$

Und hierbei gilt:

$$\mathcal{T}(y) \;=\; \{\; x_{c,y} \mid c \in C \;\}\;.$$

Weitere Vereinfachungen können nur vorgenommen werden, indem die Grammatikregeln $r \in G_{\text{cluster}}$ einzeln betrachtet werden.

Für die Grammatikregel $r \;=\; S \to Y_1^c \ldots Y_d^c$ mit festem $c \in C$ folgt:

$$f_r(x) = \begin{cases} 1 & \text{für } x = x_{c,y} \\ 0 & \text{sonst} \end{cases}$$

Für $A = \text{lhs}(r) = S$ gilt:

$$f_A(x) = 1\;.$$

Somit:

$$\begin{aligned} \hat{p}(r) &= \frac{\sum_{y\in\mathcal{Y}} f(y)\cdot p(x_{c,y}|y)}{\sum_{y\in\mathcal{Y}} f(y) \sum_{x\in\mathcal{T}(y)} p(x|y)} \\ &= \frac{\sum_{y\in\mathcal{Y}} f_c(y)}{\sum_{y\in\mathcal{Y}} f(y)} \\ &= \frac{|f_c|}{|f|}\;. \end{aligned}$$

Für die Grammatikregel $r \;=\; Y_i^c \to y_i$ mit festen $c \in C,\; y_i \in \mathcal{Y}_i$ folgt hingegen:

$$f_r(x) = \begin{cases} 1 & \text{für } x = x_{c,y} \text{ und } y \in \mathcal{H}(y_i) \\ 0 & \text{sonst} \end{cases}$$

Für $A = \text{lhs}(r) = Y_i^c$ gilt:

$$f_A(x) = \begin{cases} 1 & \text{für } x = x_{c,y} \\ 0 & \text{sonst} \end{cases}$$

Somit:

$$\begin{aligned} \hat{p}(r) &= \frac{\sum_{y\in\mathcal{H}(y_i)} f(y)\cdot p(x_{c,y}|y)}{\sum_{y\in\mathcal{Y}} f(y)\cdot p(x_{c,y}|y)} \\ &= \frac{\sum_{y\in\mathcal{H}(y_i)} f_c(y)}{\sum_{y\in\mathcal{Y}} f_c(y)} \\ &= \frac{f_c(y_i)}{|f_c|}\;. \qquad \textbf{q.e.d.} \end{aligned}$$

Somit sind die Re-Estimierungsformeln des EM-basierten Klassifizierungsverfahrens bewiesen, die in dem Lemma aus Abschnitt 4.1 angegeben wurden.

Mit dem Theorem aus Abschnitt 3.8 folgt, dass das EM-basierte Klassifizierungsverfahren die Log-Likelihood des gegebenen Korpus der mehrdimensionalen Datentypen während der Re-Estimierung monoton erhöht. Natürlich ist auch die Frage interessant, ob die Anwendung des EM-Algorithmus auf G_{cluster} sogar eine Parameterfolge erzeugt, die gegen ein globales Maximum der Log-Likelihood konvergiert. Gemäss dem Theorem aus Abschnitt 3.8 muss hierfür geklärt werden, ob zu jeder Grammatikkategorie $A \in G_{\text{cluster}}$ eine Konstante $\lambda \in \mathbb{R}$ existiert, sodass während aller Re-Estimierungsschritte gilt, dass:

$$\hat{p}[\ f_A\ ] > \lambda > 0\ .$$

Dem eben geführten Beweis entnimmt man, dass nach jedem Re-Estimierungsschritt die folgenden erwarteten Vorkommenshäufigkeiten der Grammatikkategorien vorliegen:

$$\hat{p}[\ f_A\ ] = \begin{cases} |f| & \text{für } A = S \\ |f_c| & \text{für } A = Y_i^c \end{cases}$$

Somit folgt mit

$$\lambda := \min_{c \in C}\ |f_c|$$

dass eine Voraussetzung für die gewünschte Konvergenzeigenschaft erfüllt ist:

$$\min_{c \in C}\ |f_c|\ >\ 0\ .$$

Die re-estimierten Parameter konvergieren daher gegen ein globales Maximum der Log--Likelihood, wenn zusätzlich garantiert werden kann, dass im EM-basierten Klassifizierungsverfahren auch die erwartete Cross-Entropie der Analysen kontrolliert werden kann.

Induktion von glatten Wahrscheinlichkeitsverteilungen für mehrdimensionale Datentypen mit Hilfe einer Invarianz-Eigenschaft des EM-basierten Klassifizierungsverfahrens

Für einen gegebenen Trainingskorpus, könnten *unbekannte Daten* als die Menge derjenigen Datentypen definiert werden, die nicht in diesem Traingskorpus vorkommen, aber in der Vereinigung aller Evaluierungskorpora, die für irgendeine Anwendung sinnvoll erscheinen. Mit dieser Definition werden die realen Daten dieser Welt in zwei Teile zerlegt: (i) die riesige Menge unbekannter Daten und (ii) die winzige Menge von Trainingsdaten. Aus diesem Grund ist die wichtigste Eigenschaft aller Wahrscheinlichkeitsmodelle ihre Fähigkeit mit unbekannten Daten umzugehen, d.h. unbekannten (aber sinnvollen) Datentypen eine psitive Wahrscheinlichkeit zuzuweisen. Da die Wahrscheinlichkeitsparamter der Modelle allein unter Berücksichtigung der Datentypen im Trainingskorpus estimiert

$\tilde{p}(y_1, y_2)$	smile	laugh	increase	fall	$\tilde{p}(y_1)$
man	0.2				0.2
woman	0.2	0.1			0.3
number			0.1		0.1
price			0.3	0.1	0.4
$\tilde{p}(y_2)$	0.4	0.1	0.4	0.1	

$p(y_1, y_2)$	smile	laugh	increase	fall	$p(y_1)$
man	0.16	0.04			0.2
woman	0.24	0.06			0.3
number			0.08	0.02	0.1
price			0.32	0.08	0.4
$p(y_2)$	0.4	0.1	0.4	0.1	

Abbildung 4.3: Invariante Randverteilungen der empirischen Wahrscheinlichkeitsverteilung (obere Tabelle) und der glatten Wahrscheinlichkeitsverteilung eines Modells (untere Tabelle) zwei-dimensionaler Datentypen

werden, wird im allgemeinen akzeptiert, dass Estimierungen für unbekannte Daten unzuverlässig sind. Im Gegensatz zu dieser allgemeinen Situation soll in diesem Abschnitt eine *Invarianz-Eigenschaft* des EM-basierten Klassifizierungsverfahrens für mehrdimensionale Datentypen vorgestellt werden, die die Annahme erhärtet, dass diese Modelle in der Lage ist, zuverlässige Estimierungen für unbekannte Datentypen zu liefern.

Die obere Tabelle in Abbildung 4.3 zeigt die empirische Wahrscheinlichkeitsverteilung

$$\tilde{p}(y_1, y_2) = |f|^{-1} \cdot f(y_1, y_2)$$

eines Spielzeugkorpus von zwei-dimensionalen Verb-Nomen-Datentypen, sowie ihre zwei *Randverteilungen* $\tilde{p}(y_1)$ und $\tilde{p}(y_2)$.

Die empirische Wahrscheinlichkeit von 0.1 in der dritten Spalte und dritten Zeile dieser Tabelle gibt an, dass 10 % der Datentoken in diesem Korpus mit dem Datentyp *(woman,laugh)* übereinstimmen, während der leere Eintrag in der dritten Spalte und zweiten Zeile angibt, dass der Datentyp *(man, laugh)* die empirische Wahrscheinlichkeit Null hat und nicht im Korpus vorkommt. Wird dieser Korpus als Trainingskorpus benutzt, ist das Paar *(woman, laugh)* ein *bekannter Datentyp*, während das Paar *(man, laugh)* ein unbekannter Datentyp ist. Aus der Definition folgt, dass unbekannte Datentypen eine empirische Wahrscheinlichkeit von Null haben. Aber es soll darauf hingewiesen werden, dass in der Tabelle auch sinnlose Datentypen vorkommen, zum Beispiel *(number, smile)*,

class 0 0.5	woman 0.6 man 0.4	smile 0.8 laugh 0.2
class 1 0.5	price 0.8 number 0.2	increase 0.8 fall 0.2

Abbildung 4.4: Mit dem EM-basierten Klassifizierungsverfahren trainiertes Spielzeugmodell zwei-dimensionaler Datentypen

die ebenfalls eine empirische Wahrscheinlichkeit von Null besitzen. In der Tabelle werden auch die zwei Randverteilungen $\tilde{p}(y_1)$ und $\tilde{p}(y_2)$ gezeigt. Der Wert 0.4 in der zweiten Spalte und sechsten Zeile heisst, dass *smile* in 40% aller Datentoken dieses Korpus vorkommt, während der Wert 0.3 in der sechsten Spalte und dritten Zeile angibt, dass *woman* in 30% vorkommt.

Die untere Tabelle in Abbildung 4.3 zeigt eine sogenannte *glatte Wahrscheinlichkeitsverteilung* dieses Korpus an. Im Idealfall weisen glatte Wahrscheinlichkeitsverteilungen den unbekannten Datentypen, in diesem Fall *(man,laugh)* und *(number, fall)*, eine positive Wahrscheinlichkeit (von 0.04 bzw. 0.02) zu, während sinnlose Datentypen ihre Null-Wahrscheinlichkeit behalten. Natürlich ist die gezeigte Wahrscheinlichkeitsverteilung nur ein einziges Beispiel aus der Vielfalt aller möglichen glatten Wahrscheinlichkeitsverteilungen. Ein schärferer Blick auf Abbildung 4.3 zeigt aber, dass ihre Randverteilungen $p(y_1)$ und $p(y_2)$ identisch mit den Randverteilungen $\tilde{p}(y_1)$ und $\tilde{p}(y_2)$ der empirischen Randverteilung sind. Diese Eigenschaft wird im folgenden als *Invarianz-Eigenschaft* bezeichnet. Invarianz-Eigenschaften sind in solchen Fällen sehr nützlich, in denen ein gegebenes Objekt leicht modifiziert werden muss, da sie den Modifikationsprozess dazu zwingen, gewisse Nebenbedingungen zu respektieren. In der eben beschriebenen Situation sollte zum Beispiel die empirische Wahrscheinlichkeitsverteilung $\tilde{p}(.)$ in eine glatte Wahrscheinlichkeitsverteilung $p(.)$ transformiert werden, und die Korrekturen werden als minimal angesehen, da die Randverteilungen invariant bleiben.

Abbildung 4.4 zeigt die zwei Klassen eines mit dem EM-basierten Klassifizierungsverfahren trainierten Modells zwei-dimensionaler Datentypen. Das Modell wurde in zehn Iterationen und mit zufällig initialisierten Startparametern trainiert. Die erste Spalte zeigt den Klassenindex und die Klassen-Wahrscheinlichkeit an. Die Nomen und ihre Wahrscheinlichkeiten sind in absteigender Ordnung in der zweiten Spalte gelistet, während die Verben in der dritten Spalte gezeigt werden. Es kann leicht gezeigt werden, dass die glatte Wahrscheinlichkeitsverteilung aus der unteren Tabelle in Abbildung 4.3 aus diesem Modell berechnet werden kann. Diese Beobachtung führt zu der Vermutung, dass mit dem EM-basierten

Klassifizierungsverfahren Wahrscheinlichkeitsverteilungen mehrdimensionaler Datentypen trainiert werden, die die zusätzliche interessante Eigenschaft besitzen, invariante Randverteilungen aufzuweisen. In den folgenden zwei Schritten wird gezeigt, dass dies in der Tat richtig ist. Die folgende Formel zeigt zunächst, dass die i-te Randverteilung $p(y_i)$ eines mit dem EM-basierten Klassifizierungsverfahren trainierten Modells sehr einfach aus den Klassenwahrscheinlichkeiten und den bedingten Wahrscheinlichkeiten für y_i gegeben einer Klasse berechnet werden kann:

$$\begin{aligned}
p(y_i) &= \sum_{y \in \mathcal{H}(y_i)} p(y) \\
&= \sum_{y \in \mathcal{H}(y_i)} \sum_{c \in C} p(c) \prod_{j=1}^{d} p(y_j|c) \\
&= \sum_{c \in C} p(c) \sum_{y \in \mathcal{H}(y_i)} \prod_{j=1}^{d} p(y_j|c) \\
&= \sum_{c \in C} p(c) \cdot p(y_i|c) \prod_{j \neq i} \sum_{y_j \in \mathcal{Y}_j} p(y_j|c) \\
&= \sum_{c \in C} p(c) \cdot p(y_i|c) \ .
\end{aligned}$$

Es ist interessant, dass in der gezeigten Berechnung lediglich die Modell-Eigenschaften des EM-basierten Klassifizierungsverfahrens benötigt wurden. In dem zweiten Schritt werden auch die Re-Estimierungsformeln benutzt. Es gilt:

$$\begin{aligned}
\hat{p}(y_i) &= \sum_{c \in C} \hat{p}(c) \cdot \hat{p}(y_i|c) \\
&= \sum_{c \in C} |f|^{-1} \cdot |f_c| \cdot |f_c|^{-1} \cdot f_c(y_i) \\
&= \sum_{c \in C} |f|^{-1} \sum_{y \in \mathcal{H}(y_i)} f_c(y) \\
&= \sum_{y \in \mathcal{H}(y_i)} |f|^{-1} \cdot f(y) \sum_{c \in C} p(c|y) \\
&= \sum_{y \in \mathcal{H}(y_i)} \tilde{p}(y) \\
&= \tilde{p}(y_i) \ .
\end{aligned}$$

Dieses Resultat zeigt, dass die Randverteilungen während des gesamten Re-Estimierungsprozesses invariant sind:

$$p(y_i) = \tilde{p}(y_i) \qquad (i = 1 \ldots d) \ .$$

Es ist wichtig, festzuhalten, dass die EM-basierten Re-Estimierungsformeln aus Abschnitt 4.1 als Schlüsselschritt benutzt wurden, um dieses Resultat herzuleiten. Aus diesem

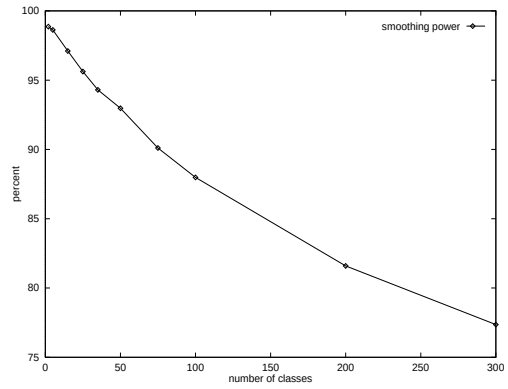


Abbildung 4.5: Evaluierung der Glättungskraft

Grund kann vermutet werden, dass andere statistische Inferenz-Methoden nicht dieses gute Resultat erzielen, selbst wenn sie auf dasselbe Klassifizierungsmodell angewendet werden.

Durch die Entdeckung der vorgestellten Invarianz-Eigenschaften ermutigt, wurden die mit Hilfe des EM-basierten Klassifizierungsverfahrens trainierten Modelle einer Evaluierung unterzogen, um ihre Glättungs-Eigenschaften auch experimentell nachzuweisen. Die sogenannte *Glättungskraft* eines Modells für mehrdimensionale Datentypen ist als Anzahl aller Datentypen definiert, die von dem Modell eine positive Wahrscheinlichkeit zugewiesen bekommen, wobei diese Zahl normalisiert wird, indem durch die Grösse aller möglichen (auch der sinnlosen) mehr-dimensionalen Datentypen, d.h. durch die Grösse der Menge $\mathcal{Y}_1 \times \ldots \times \mathcal{Y}_d$ dividiert wird. Unter Benutzung der Invarianz-Eigenschaft folgt für die mit dem EM-basiertem Klassifiaktionsverfahren trainierten Modelle mit genau einer Klasse ($|C| = 1$), dass:

$$p(y) = \prod_{i=1}^{d} p(y_i) = \prod_{i=1}^{d} \tilde{p}(y_i) \ .$$

Offensichtlich haben diese Modelle also eine maximale Glättungskraft von 100%. Aber es kann erwartet werden, dass die Glättungskraft dieser Modelle sinkt, wenn die Zahl der Klassen auf sinnvollere Werte ansteigt, da im Grenzwert

$$p \to \tilde{p} \quad (|C| \to \infty)$$

gilt. Abbildung 4.5 zeigt die Ergebnisse für die Glättungskraft von einigen mit dem EM-basierten Klassifikationsverfahren (mit konstant 50 Iterationen) trainierten Modellen zweidimensionaler Verb-Nomen-Datentypen (siehe Abschnitt 4.3). Zum Beispiel weist ein Modell mit 35 Klassen eine Glättungskraft von ungefähr 95% auf, was ungefähr 700-mal besser ist als die Glättungskraft der empirischen Wahrscheinlichkeitsverteilung, die eine Glättungskraft von minimalen 0.14% besitzt. Startparameter haben auf die Evaluierungsergebnisse natürlich einen Einfluss, der aber nur bei ungefähr 1% liegt.

Eine neue Pruning-Technik auf der Grundlage klassenbasierter Häufigkeiten mehrdimensionaler Datentypen

Es ist normalerweise der Fall, dass mehr Trainingsdaten die mit dem EM-basierten Klassifizierungsverfahren trainierten Modelle verbessern werden. Aber natürlich wird mit der Grösse des Trainingskorpus auch die Grösse der Modelle anwachsen, was im Extremfall zu Modellen führen kann, die zu gross sind, um in der Praxis benutzt werden zu können. Um mit diesem Problem umgehen zu können, wird in diesem Abschnitt eine neue Pruning-Technik vorgeschlagen, die die seltenen Datentypen auf der Grundlage der klassenbasierten Häufigkeiten aus den Modellen entfernt.

Klassen-basierte Häufigkeiten $f_c(y)$ spielen im Kontext des EM-basierten Klassifizierungsverfahrens eine Hauptrolle:

(i) sie können sehr erfolgreich angewendet werden, um lexikalische Ambiguitäten aufzulösen (Abschnitt 4.5),

(ii) die *Randhäufigkeiten* $f_c(y_i)$ sind die Schlüsselvariablen der sehr einfachen Re-Estimierungsformeln (Abschnitt 4.1).

Konsequenterweise werden die klassenbasierten Häufigkeiten $f_c(y)$ daher auch in der Implementierung des EM-basierten Klassifikationsverfahrens benutzt (anstelle der theoretischen Modellparameter $p(c)$ und $p(y_i|c)$). Unglücklicherweise kann gezeigt werden, dass die Anzahl der internen Parameter während des mathematischen Re-Estimierungsprozesses nicht sinken wird, weil *Positivität* eine weitere Invarianz-Eigenschaft des EM-basierten Klassifikationsverfahrens ist (zumindest für jede endliche Anzahl von Iterationen). Aus diesem Grund wird ein Modell mit $|C|$ Klassen ungefähr $|C|$-mal mehr Speicherplatz als die originalen Daten benötigen. Deshalb wurde eine klassenweise *Pruning-Technik* angewendet, um die Modelle des EM-basierten Klassifikationsverfahrens während des Re-Estimierungsprozesses zu verkleinern.

Nach jedem Schritt des EM-Algorithmus werden alle seltenen klassen-basierten Häufigkeiten

$$f_c(y) < \epsilon \cdot |C|^{-1}$$

aus den Modellen entfernt. Aus Erfahrung wurde in allen bisher durchgeführten Experimenten $\epsilon = 0.001$ gewählt. Der Schwellenwert $\epsilon \cdot |C|^{-1}$ nimmt ab, wenn die Anzahl der Klassen zunimmt. Dies ist ein erwünschter Effekt, da auch die klassenbasierten Häufigkeiten $f_c(y)$ kleiner werden, wenn die Anzahl der Klassen zunimmt. Das Letztere folgt,

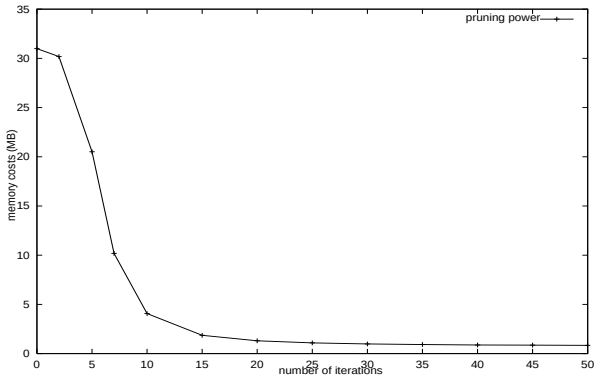


Abbildung 4.6: Evaluierung anhand der Pruning-Aufgabe

weil die empirische Häufigkeit sich auf die Klassen verteilt:

$$f(y) = \sum_{c \in C} f_c(y) \ .$$

Somit macht es Sinn, den Schwellenwert ϵ für $f_c(y) \cdot |C|$ anstelle für $f_c(y)$ zu benutzen.

Die mit dem EM-basierten Klassifikationsverfahren trainierten Modelle wurden empirisch anhand einer Pruning-Aufgabe evaluiert. Die Eingabe des EM-basierten Klassifikationsverfahrens bestand aus einem Korpus von ungefähr 27 000 Verb-Objekt-Paaren, die aus dem Penn Wall Street Journal extrahiert wurden. Abbildung 4.6 zeigt den während der Re-Estimierung für die Speicherung eines Modells erforderlichen Festplattenplatz (in MB) an. In diesem Experiment sank der erforderliche Speicherplatz dramatisch zwischen der 3ten und 10ten Iteration. Modelle, die mit mehr als 15 Iterationen trainiert wurden, benötigten nur doppelt soviel Platz wie der originale Korpus. In zukünftigen Experimenten sollen mögliche Interaktionen zwischen der Glättungskraft, der "Pruning-Kraft", sowie der linguistischen Qualität der trainierten Modelle betrachtet werden.

Class 14 PROB 0.0283		0.0877	0.0303	0.0187	0.0179	0.0161	0.0154	0.0151	0.0150	0.0130	0.0112	0.0094	0.0088	0.0086	0.0085	0.0073	0.0066	0.0066	0.0064	0.0061	0.0061	0.0059	0.0058	0.0057	0.0053	0.0051	0.0051	0.0043	0.0042	0.0041	0.0041
		Polizei	deutschLand	Nation	SPD	USA	Koalition	Sprecher	Verein	StaatsAnwaltschaft	Behörde	Bonn	firm	UNO	BundesAmt	Veranstalter	Blatt	Konzern	Magistrat	Sender	oberBürgerMeister	BundesBank	BürgerMeister	fern+sehen	Nato	Zeitung	Nähe	nachrichtenAgentur	PolizeiSprecher	LandesRegierung	rundFunk
0.0581	mit+teilen-VPA.n:NP.Nom	•						•	•		•				•	•		•	•	•			•					•	•		•
0.0220	berichten-VPA.n:NP.Nom	•			•			•			•				•		•			•		•		•		•		•	•		•
0.0141	vereinen-ADJ		•	•																												
0.0120	sagen-VPA.n:NP.Nom	•						•								•							•									
0.0114	Bremer-ADJ	•										•								•			•							•		
0.0113	mit+teilen-VPA.np:NP.Nom			•				•		•	•								•			•	•									
0.0095	erklären-VPA.n:NP.Nom	•						•	•						•						•		•						•			
0.0070	hessisch-ADJ	•			•																									•	•	
0.0065	unmittelbar-ADJ																										•					
0.0064	fordern-VPA.na:NP.Nom		•		•	•			•			•							•							•				•		
0.0063	machen-VPA.nad:NP.Nom	•			•	•			•	•									•													
0.0061	verzichten-VPA.np:NP.Nom		•		•	•	•				•			•											•					•		
0.0060	melden-VPA.n:NP.Nom							•							•									•		•		•				•
0.0059	Berliner-ADJ	•							•	•	•									•						•				•		
0.0055	spielen-VPA.nap:NP.Nom		•			•																										
0.0052	Westdeutsch-ADJ												•					•														•
0.0046	statistisch-ADJ														•																	
0.0043	meinen-VPA.n:NP.Nom					•		•											•				•									
0.0043	auf+nehmen-VPA.nap:NP.Nom	•	•		•	•				•		•		•																		
0.0042	wünschen-VPA.nar:NP.Akk																															
0.0041	vor+stellen-VPA.nar:NP.Nom				•		•		•													•			•					•		
0.0041	haben-VPA.nap:NP.Nom	•	•		•	•	•				•				•	•						•		•	•					•		
0.0041	betonen-VPA.n:NP.Nom	•														•					•		•									
0.0040	zuständig-ADJ									•	•												•									
0.0039	übernehmen-VPA.na:NP.Nom	•			•	•	•		•		•	•		•				•	•													
0.0038	niedersächsisch-ADJ	•			•																										•	
0.0036	örtlich-ADJ	•							•		•																•					
0.0035	bestätigen-VPA.n:NP.Nom							•		•	•										•		•						•			
0.0035	erzählen-VPA.n:NP.Nom							•																								
0.0033	ein+treffen-VPA.n:NP.Nom	•																								•						

Abbildung 4.7: Deutsche Klasse 14: Regierungsstellen, öffentliche Gewalt

4.3 Aufbau semantisch annotierter Lexika

In diesem Abschnitt werden Ergebnisse vorgestellt, die zum grossen Teil aus gemeinsamer Arbeit mit Franz Beil, Glenn Carroll, Stefan Riezler und Mats Rooth resultieren und teilweise in Rooth et al. (1998) und Rooth et al. (1999) veröffentlicht wurden.

Die Konstruktion sehr grosser maschinell erstellter Lexika für die vielen natürlichen Sprachen, in denen nun sehr grosse Sprachkorpora zur Verfügung stehen, ist eine der herausfordernsten Aufgaben der maschinellen Sprachverarbeitung. Resnik (1993) initiierte Forschung zur automatischen Gewinnung von semantischen Selektionsrestriktionen. Ribas (1994) präsentierte einen Ansatz, der die syntaktische Position eines Elementes berücksichtigte, dessen semantische Relation bestimmt werden sollte. Leider benötigen beide Ansätze und auch die meisten danach folgenden Ansätze eine fixierte semantische Taxo-

nomie als Resource. Dies ist ein Problem, weil (i) solche semantische Hierarchien zur Zeit nur für die wenigsten Sprachen zur Verfügung stehen, und (ii) es eine noch offene Frage ist, ob und zu welchem Grad eine fixierte semantische Hierarchie dafür geeignet ist, lexikalische Bedeutung zu repräsentieren. Beide Betrachtungen legen nahe, dass induktive und experimentelle Ansätze zur Konstruktion semantischer Hierarchien nötig sind.

Dieser Abschnitt präsentiert eine zweimalige Anwendung des EM-Algorithmus, in Form (i) einer Methode zur Konstruktion von Klassifikationsmodellen für Verb-Nomen-Paare, sowie (ii) einer Methode zur maschinellen Induktion von semantisch annotierten Subkategorisierungsrahmen. Hierbei werden die im ersten Schritt erstellten Klassifikationsmodelle benutzt.

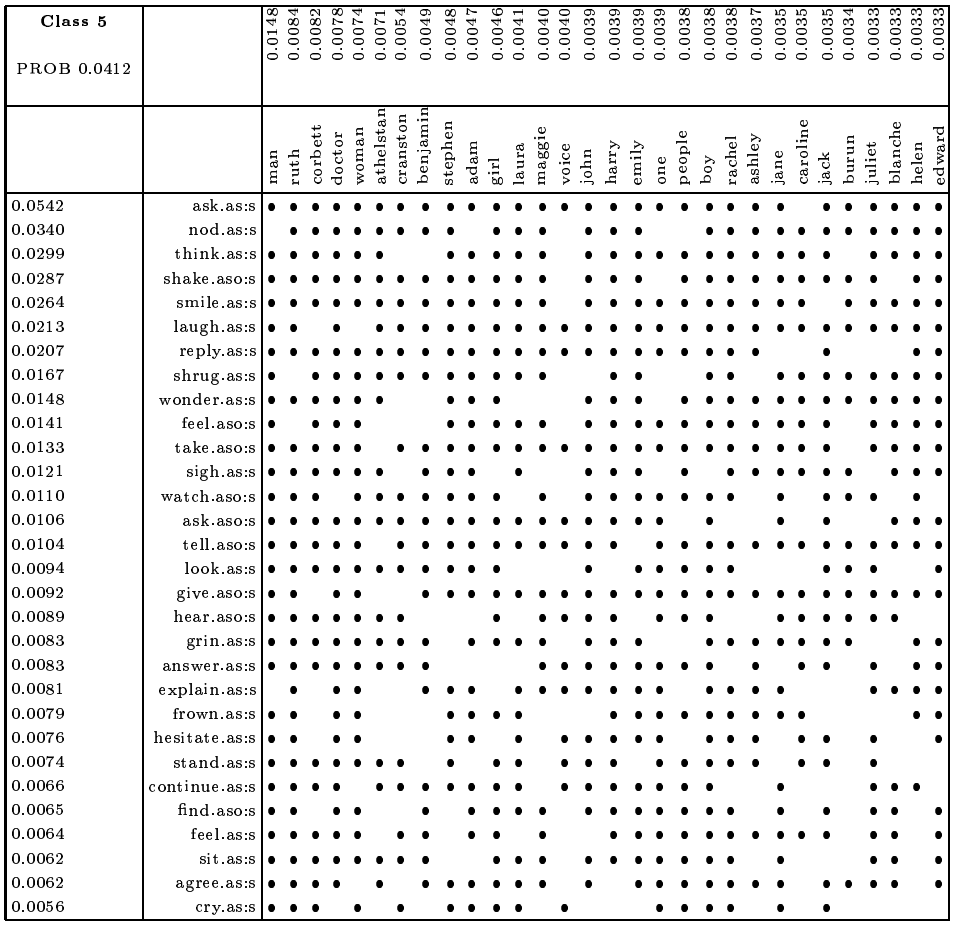


Abbildung 4.8: Englische Klasse 5: Kommunikative Aktion

EM-basierte Klassifikationsverfahren zwei-dimensionaler Datentypen

Die Schlüsselidee des EM-basierten Klassifikationsverfahrens zwei-dimensionaler Datentypen wurde in Rooth (1995) präsentiert. Eine wichtige Eigenschaft dieses Ansatzes ist, dass er eine "weiche Klassifikationsmethode" darstellt, welche die Klassenmitgliedschaft mit Hilfe einer Wahrscheinlichkeitsverteilung über Verben und Nomen ausdrückt. Im Gegensatz dazu sind die "harten Klassifikationsmethoden", wie die von Brown et al. (1992), wo jedes Wort zu genau einer Klasse gehört, wegen *Homonymie* eher unrealistisch. Ferner hebt sich dieser Ansatz durch die Verwendung reiner Wahrscheinlichkeitsverteilungen wohltuend gegenüber Methoden ab, die lediglich heuristische Ähnlichkeitsmasse benutzen (Dagan et al. 1998). Gegenwärtig aktuelle Klassifikationsmethoden ((Saul und Pereira 1997), (Hofmann und Puzicha 1998)) benutzen zwar ebenfalls reine Wahrscheinlichkeitsverteilungen, aber nicht den EM-Algorithmus, sondern alternative statistische Inferenzmethoden. Ferner werden dort die trainierten Modelle für einfachere Aufgaben benutzt, die keine Kombination des EM-Algorithmus erfordern, wie in dem folgenden Experiment.

Das EM-basierte Klassifikationsverfahren wurde für den mehrdimensionalen Fall bereits in Abschnitt 4.1 vorgestellt. Da der zweidimensionale Fall ein einfacher, aber bedeutender Spezialfall ist, soll das EM-basierte Klassifikationsverfahren an dieser Stelle auch in dieser speziellen Form vorgestellt werden. Für $d = 2$ werden die Bezeichnungen

$$V = \mathcal{Y}_1, \;\; N = \mathcal{Y}_2, \;\; \text{und } (v, n) = (y_1, y_2)$$

benutzt. Damit hat das *EM-basierte Klassifikationsmodell für Verb-Nomen-Paare* das folgende Aussehen:

$$\begin{aligned} p(v,n) &= \sum_{c \in C} p(c,v,n) \\ &= \sum_{c \in C} p(c) \cdot p(v,n|c) \\ &= \sum_{c \in C} p(c) \cdot p(v|c) \cdot p(n|c) \; . \end{aligned}$$

Re-Estimierungsformeln für das EM-basierte Klassifikationsmodell für Verb-Nomen-Paare: Für die re-estimierten Parameter $\hat{p}(c)$, $\hat{p}(v|c)$, $\hat{p}(n|c)$ gilt ($c \in C, \; v \in V, \; n \in N$):

$$\begin{aligned} \hat{p}(c) &= |f|^{-1} \cdot |f_c| \; , \\ \hat{p}(v|c) &= |f_c|^{-1} \cdot f_c(v) \; , \\ \hat{p}(n|c) &= |f_c|^{-1} \cdot f_c(n) \; . \end{aligned}$$

Hierbei sind $f_c(v)$ und $f_c(n)$ die *Marginalhäufigkeiten* der *klasssenbasierten Häufigkeiten*

$$f_c(v,n) = p(c|v,n) \cdot f(v,n) \; ,$$

die sich einerseits aus dem Trainingskorpus $f(v,n)$, andererseits aus den gegenwärtigen Parametern $p(c)$, $p(v|c)$, $p(n|c)$ berechnen.

Gemäss dem Beweis in Abschnitt 4.2 gilt, dass die Log-Likelihood des gegebenen Trainingskorpus von Verb-Nomen-Datentypen während des Trainings mononton anwächst und (bei kontrollierbarer erwarteter Cross-Entropie der Analysen) sogar *ein globales Maximum* annehmen wird.

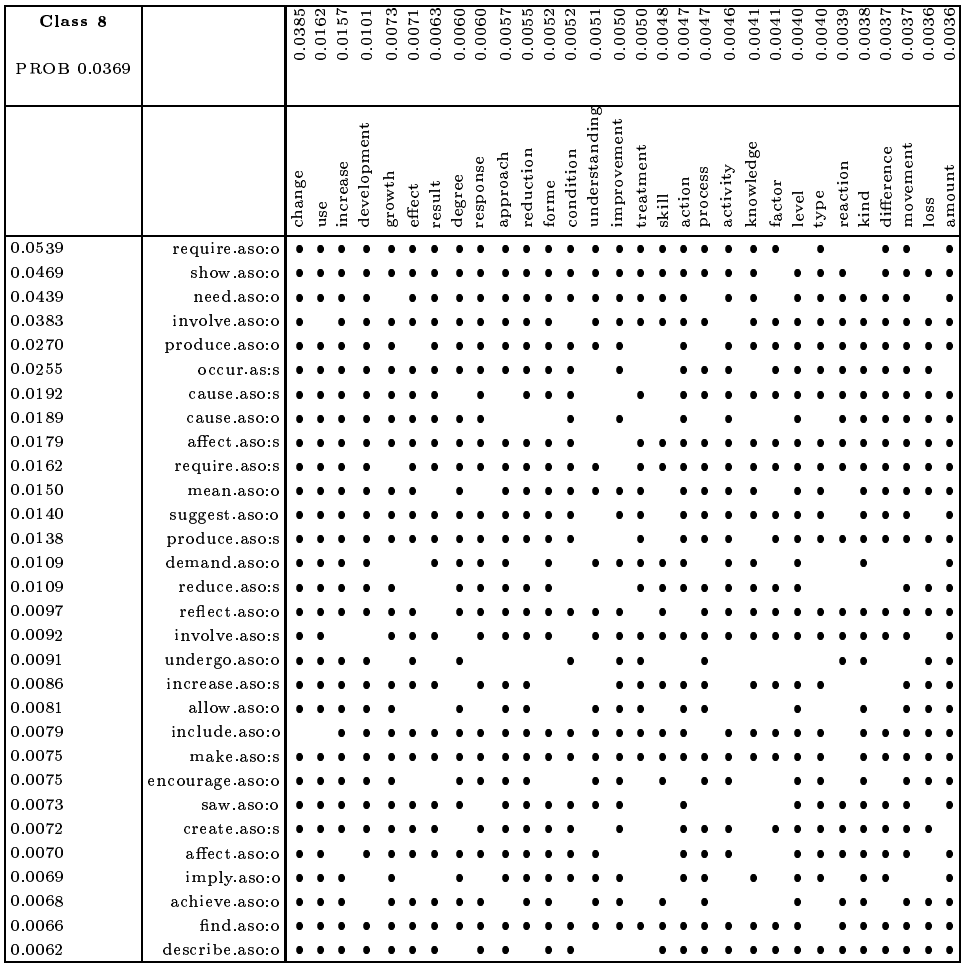


Abbildung 4.9: Englische Klasse 8: Dispositionen

Illustrative Beispiele induzierter Verb-Argument-Klassen

Im folgenden sollen einige Beispiele von induzierten Verb-Argument-Klassen vorgestellt werden. Die Eingabe des EM-basierten Klassifikationsverfahren war in einem Experiment ein Korpus von 1 178 698 Datentoken (608 850 Datentypen) von Verb-Nomen-Paaren, die als lexikalische Köpfe aus den grammatischen Relationen intransitiver und transitiver Verben und ihrer Subjekte und Objekte extrahiert wurden. Die Daten wurden aus den wahrscheinlichsten Synatxbäumen extrahiert, die die lexikalisierte probabilistische kontextfreie Grammatik von Carroll und Rooth (1998) angewandt auf den *British National Corpus* (BNC) gab (117 Millione Datentoken).

Abbildung 4.8 zeigt eine induzierte Klasse aus einem Modell mit 35 Klassen. Oben werden die 30 wahrscheinlichsten Nomen der $p(n|5)$-Wahrscheinlichkeitsverteilung zusammen mit ihren Wahrscheinlichkeiten gezeigt, während links die 30 wahrscheinlichsten Verben der $p(v|5)$-Wahrscheinlichkeitsverteilung gezeigt werden, wobei 5 der Klassenindex ist. So-

Class 17 PROB 0.0265		0.0379	0.0315	0.0313	0.0249	0.0164	0.0143	0.0110	0.0109	0.0105	0.0103	0.0099	0.0091	0.0089	0.0088	0.0082	0.0077	0.0073	0.0071	0.0071	0.0070	0.0068	0.0067	0.0065	0.0065	0.0058	0.0057	0.0057	0.0054	0.0051	0.0050
		number	rate	price	cost	level	amount	sale	value	interest	demand	chance	standard	share	risk	profit	pressure	income	performance	benefit	size	population	proportion	temperature	tax	fee	time	power	quality	supplely	money
0.0437	increase.as:s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
0.0392	increase.aso:o		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●		●	●	●	●	●
0.0344	fall.as:s	●	●	●	●	●	●	●	●		●	●	●	●		●	●	●	●			●	●	●	●		●		●	●	
0.0337	pay.aso:o	●	●	●		●	●		●	●	●			●				●	●	●			●				●	●			●
0.0329	reduce.aso:o	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
0.0257	rise.as:s	●	●	●	●	●	●	●	●		●		●	●	●	●	●	●		●	●	●	●	●	●		●	●	●	●	
0.0196	exceed.aso:o	●	●	●	●	●	●		●	●	●	●	●			●		●		●	●	●	●		●	●	●	●	●	●	
0.0177	exceed.aso:s	●	●	●	●	●	●	●	●	●	●			●	●	●	●	●	●	●	●	●	●	●	●		●			●	●
0.0169	affect.aso:o	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●		●	●	●	●	●	
0.0156	grow.as:s	●						●	●	●	●			●	●	●	●	●		●	●	●	●					●			●
0.0134	include.aso:s	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	●	●	●		●	●			●	●	●	●	●	●
0.0129	reach.aso:s	●	●	●	●	●	●	●		●	●			●		●	●		●	●		●	●	●			●	●	●	●	●
0.0120	decline.as:s	●	●	●	●	●		●	●	●	●		●	●	●	●					●	●	●	●					●	●	
0.0102	lose.aso:o	●	●		●		●	●	●	●		●	●	●	●	●	●	●		●		●	●	●		●	●	●	●		●
0.0099	act.aso:s		●	●				●					●							●					●			●		●	
0.0099	improve.aso:o	●	●			●			●			●	●	●		●			●								●	●	●		
0.0088	include.aso:o	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
0.0088	cut.aso:o	●	●	●	●	●		●	●			●	●		●	●		●		●					●	●	●				
0.0080	show.aso:o	●	●		●	●	●	●	●	●	●		●	●	●	●	●	●	●	●	●	●					●	●	●		
0.0078	vary.as:s	●	●	●	●	●	●		●	●	●		●						●		●		●	●			●		●		
0.0072	give.aso:o	●	●	●	●	●	●		●	●		●	●	●	●	●	●	●	●	●	●	●	●	●			●	●	●		●
0.0071	carry.aso:o	●	●	●	●	●	●			●			●		●			●		●			●	●		●	●		●		●
0.0068	improve.as:s	●	●	●		●		●	●			●	●			●			●								●	●	●		
0.0066	have.as:s	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	●	●	●		●	●	●	●	●	●	●
0.0066	produce.aso:o	●	●		●	●	●	●	●	●	●			●	●	●	●	●	●	●	●		●				●	●	●	●	●
0.0066	get.aso:o	●	●	●	●	●	●	●	●	●	●			●	●		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
0.0064	raise.aso:o	●		●	●	●	●	●		●	●	●	●	●	●	●	●	●	●	●				●	●				●	●	●
0.0063	mean.aso:o	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●		●		●	●	●	●	●
0.0062	receive.aso:o	●		●	●	●	●		●	●	●	●	●	●		●		●	●	●		●			●	●	●	●	●	●	●
0.0058	stand.aso:o		●									●			●																●

Abbildung 4.10: Englische Klasse 17: Skalare Änderungen

che Verb-Nomen-Paare, die im Trainingskorpus vorkommen, werden in der Matrix mit einem fetten Punkt (•) markiert. Verben mit Suffix *as:s* markieren das Subjektargument eines aktiven intransitiven Verbs, während analog dazu *aso:s* und *aso:o* das Subjekt- bzw. Objektargument eines aktiven transitiven Verbs notieren. Somit besteht v in der obigen Diskussion aus der Kombination eines Verbes mit einem Argument seines *Subkategorisierungsrahmens as.s, aso:s* oder *aso:o*. Induzierte Klassen haben oft eine lexikalische Semantik: Klasse 5 ist eine Ansammlung von Verb-Nomen-Entitäten, wobei alle Nomen Personen bezeichnen und die Verben eine Art *kommunikative Aktion*. Abbildung 4.10 zeigt einen Haufen von Verben, die eine *skalare Änderung* anzeigen, sowie Dinge, die sich entlang gewisser Skalen bewegen. Abbildung 4.9 kann als Klasse von *Dispositionen* mit verschiedenen Ausführungsmodi interpretiert werden.

In einem anderen Experiment wurden 418 290 Datentoken(318 086 Datentypen) deutscher Verb-Nomen- bzw. Adjektiv-Nomen-Paare benutzt, die ebenfalls aus den grammatischen Relationen von Syntaxbäumen maximaler Wahrscheinlichkeit extrahiert wurden. Zu diesem Zeck wurde ein 4,1 Millionen-Wort-Korpus, bestehend aus ca. 450 000 Deutschen Verbletzt-Sätzen geparst. Das hierfür benutzte lexikalisierte statistische Modell wurde in Beil et al. (1998) beschrieben.

Die Abbildungen 4.7 und 4.11 zeigen zwei Klassen aus einem Modell mit 35 Klassen. Oben und links werden die 30 wahrscheinlichsten Verb/Adjektiv-Prädikate und ihre lexikalischen Argumente gezeigt, erneut absteigend nach ihrer Wahrscheinlichkeit in der gegebenen Klasse sortiert. Verbale Prädikate werden mit Subkategorisierungsrahmen, sowie einem der Argumente notiert. Zum Beispiel notiert *liegen-VPA.np:NP.Nom* das Subjekt (*:NP.Nom*) des aktiven Verbs (*-VPA*) *liegen*, welches eine nominative und präpositionale Phrase subkategorisiert (*.np*). Analog notiert *tragen-VPA.na:NP.Akk* das direkte Objekt (*NP.Akk*) des aktiven tranistiven Verbs (*-VPA.na*) *tragen* und *steigen-VPA.n:NP.Nom* das Subjekt des intransitiven Verbs *steigen*. Eine erste Beobachtung ist, dass die deutschen Klassen im Vergleich zu den englischen Klassen wesentlich spärlicher besetzt sind, was natürlich deswegen erklärlich ist, weil in diesem Experiment ein sehr viel kleinerer Trainingskorpus benutzt wurde.

Abbildung 4.7 zeigt eine Klasse, die als *Regierungsstelle, öffentliche Gewalt* interpretiert werden kann, während Abbildung 4.11 ebenfalls eine KLasse der *skalaren Änderung* zeigt (das Gegenstück zur englischen Klasse 17).

Class 26 PROB 0.0223		0.0379	0.0226	0.0182	0.0171	0.0137	0.0133	0.0129	0.0086	0.0079	0.0079	0.0073	0.0072	0.0071	0.0067	0.0064	0.0063	0.0061	0.0057	0.0056	0.0053	0.0051	0.0047	0.0046	0.0039	0.0038	0.0037	0.0037	0.0036	0.0036	0.0035
		Ergebnis	Preis	Menge	Anteil	Stück	Zahl	Gewinn	Kritiker	BürgerMeister	Angst	Umsatz	Einnahme	Zins	Schicksal	Bus	NachFrage	Ertrag	Figur	Verlust	Frucht	ArbeitsLosigkeit	Kost	Last	WahlErgebnis	Temperatur	Solidarität	InflationsRate	Abschluß	Import	unterSchrift
0.0601	liegen-VPA.np:NP.Nom	•	•	•	•	•	•	•		•				•	•			•		•			•			•		•			
0.0351	tragen-VPA.na:NP.Akk			•	•									•						•	•		•	•							
0.0230	steigen-VPA.n:NP.Nom		•	•	•		•					•	•	•			•			•		•				•		•			
0.0213	sagen-VPA.n:NP.Nom								•	•																					
0.0182	gering-ADJ		•	•	•		•						•	•				•		•		•	•			•		•			
0.0135	sinken-VPA.n:NP.Nom		•		•			•					•	•			•	•									•				
0.0135	steigen-VPA.np:NP.Nom				•		•										•	•		•			•				•				
0.0119	erklären-VPA.n:NP.Nom									•																					
0.0100	positiv-ADJ	•																•													
0.0091	zurück+gehen-VPA.np:NP.Nom				•		•	•				•	•					•										•		•	
0.0087	sinken-VPA.np:NP.Nom				•		•					•		•												•	•			•	
0.0082	spät-ADJ												•					•													
0.0077	wirken-VPA.n:NP.Nom																														
0.0071	ändern-VPA.na:NP.Akk														•																
0.0071	regional-ADJ																														
0.0067	Erfolgreich-ADJ																												•		
0.0065	bestätigen-VPA.n:NP.Nom									•	•																				
0.0057	steigen-ADJ		•				•	•				•		•			•						•	•				•			
0.0056	an+steigen-VPA.np:NP.Nom		•		•		•					•											•			•		•			
0.0054	formulieren-VPA.na:NP.Nom									•																					
0.0052	böse-ADJ														•																
0.0051	an+steigen-VPA.n:NP.Nom				•			•									•						•	•			•				
0.0051	sitzen-VPA.n:NP.Nom										•																				
0.0049	übersteigen-VPA.na:NP.Nom	•	•		•		•					•	•	•			•			•			•					•			
0.0045	ein+setzen-VPP.n:NP.Nom															•															
0.0045	an+erkennen-VPA.na:NP.Akk	•																					•		•				•		
0.0045	entdecken-VPA.nap:NP.Akk										•								•												
0.0043	zu+nehmen-VPA.np:NP.Nom	•			•						•	•	•				•						•							•	
0.0043	betrachten-VPA.na:NP.Akk						•																								
0.0043	senken-VPP.n:NP.Nom		•	•			•							•			•											•			

Abbildung 4.11: Deutsche Klasse 26: skalare Änderung

Pseudo-Disambiguierung

Die englischen EM-basierten Klassifikationsmodelle wurden anhand einer Pseudo-Disambiguierungsaufgabe evaluiert, die der in Pereira et al. (1993) ähnelt, aber im Detail Unterschiede aufweist. Die Aufgabe ist es, zu entscheiden, welches von zwei Verben v und v' ein gegebenes Nomen n mit grösserer Wahrscheinlichkeit als Argument nimmt. Hierzu wurde das Paar (v, n) dem originalen Korpus entnommen, während das Paar (v', n) auf trickreiche Weise so konstruiert wurde, dass (i) für einen gegebenen Trainingskorpus ein ungesehener Datentyp (v', n) resultiert, und (ii) das Verb-Nomen-Paar (v', n) gegenüber dem Paar (v, n) eine intuitiv weniger wahrscheinliche Kombination darstellt. Somit kann in dieser Evaluierung getestet werden, wie gut die EM-basierten Klassifikationsmodelle über *ungesehene Verben generalisieren* werden.

Die Daten wurden folgendermassen konstruiert. Zunächst wurde ein sogenannter *Testkorpus* von 3 000 (v, n)-Paaren per Zufallsgenerator aus dem originalen Korpus von 1 280 712 Datentoken ausgewählt. Anschliessend wurden aus dem originalen Korpus alle Datentoken entfernt, die mit einem der Datentypen aus dem Testkorpus übereinstimmten. Dies ergab den *Trainingskorpus* von 1 178 698 Datentoken. Danach wurde aus dem Testkorpus ein Evaluierunskorpus von (v, n, v')-Tripeln konstruiert, indem für jedes Verb-Nomen-Paar (v, n) aus dem Testkorpus, ein Verb v' zufällig (aber gemäss seiner Häufigkeit im Trainingskorpus) ausgewählt wurde, wobei das Paar (v', n) weder im Trainings- noch im Testkorpus vorkommen durfte. Aber natürlich wurde verlangt, dass v, v', und n im Trainingskorpus vorkommen. Zusätzlich wurde darauf geachtet, dass nur solche Verben und Nomen in den Evaluierungskorpus aufgenommen wurden, die mindestens 30-mal, aber höchstens 3 000-mal im Trainingskorpus vorkamen. Die resultierenden 1 337 Tripel aus dem Evaluierungskorpus wurden benutzt, um eine Folge von EM-basierten Klassifikationsmodellen zu evaluieren, die auf dem Trainingskorpus mit den in Abschnitt 4 angegebenen Re-Estimierungsformeln trainiert wurden.

Die Evaluierung der EM-basierten Klassifikationsmodelle fand unter Berücksichtigung der freien Parameter des EM-Algorithmus (Startparameter und Anzahl Trainingsiterationen) und des einen freien Parameters eines EM-basierten Klassifikationsmodells (Anzahl der Klassen) statt, was dazu führte, dass $3 \times 10 \times 6$ Modelle zu evaluieren waren.

Die *Akuratheit* eines Modells wurde gemessen, indem die Anzahl der Entscheidungen gegen das zufällige Verb $(p(n|v) \geq p(n|v'))$ gezählt und durch die Grösse des Evaluierungskorpus geteilt wurde. Abbildung 4.12 zeigt die Evaluierungsresultate für die besten englischen Modelle (trainiert mit 50 Iterationen, gemittelt über die Startwerte). Entlang der x-Achse wird die Anzahl der Klassen aufgetragen. Verschiedene Startwerte haben einen kleinen

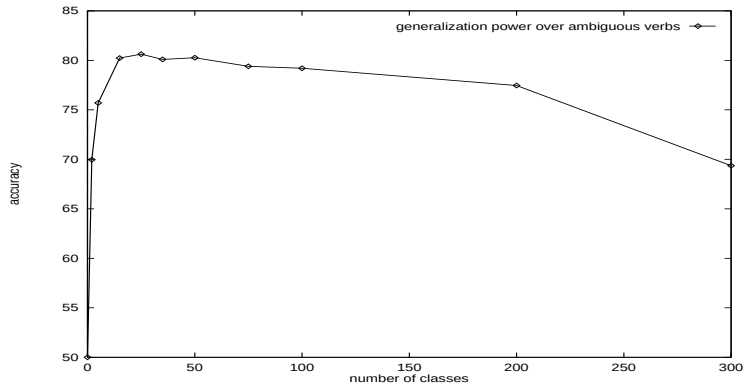


Abbildung 4.12: Pseudo-Disambiguierung der englischen Modelle

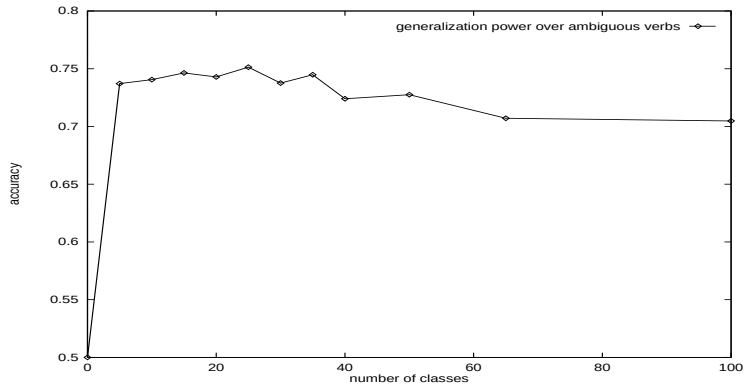


Abbildung 4.13: Pseudo-Disambiguierung der deutschen Modelle

Effekt von ca. 2%. Modelle mit 25 bis 100 Klassen weisen ungefähr 80% Akuratheit auf. Modelle mit mehr als 100 Klassen zeigen einen kleinen aber stabilen *Übertrainingseffekt.*

Die deutschen Modelle wurden analog evaluiert. Ein Evaluierungskorpus von 886 (v, n, v')-Tripeln wurde aus dem originalen Korpus extrahiert. Ein Trainingskoprpus von 418 290 Datentoken wurde zum Training einer Sequenz von $3 \times 11 \times 20$ EM-basierten Klassifikationsmodell benutzt. Abbildung 4.13 zeigt die Evaluierungsresultate für die besten Modelle (trainiert mit 100 Iterationen, gemittelt über die Startwerte). Es wurde eine Akuratheit von 75% für Modelle mit bis zu 35 Klassen gemessen. Verschiedene Startwerte hatten einen Einfluss von ca. 2% auf die Ergebnisse. Für Modelle mit mehr als 50 Klassen wird wieder ein kleiner Übertrainingseffekt beobachtet.

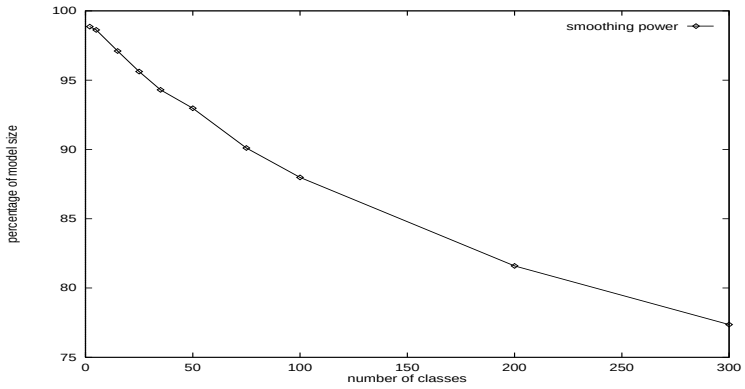


Abbildung 4.14: Glättungskraft der englischen Modelle

Evaluierung der Glättungskraft

In einem zweiten Experiment wurde die *Glättungskraft* der EM-basierten Klassifikationsmodell evaluiert. Die Glättungskraft eines Modells ist als der Prozentanteil der (v, n)-Paare im Raum $V \times N$ aller möglichen Kombinationen von Verben und Nomen definiert, die eine positive Modellwahrscheinlichkeit haben. Die Menge $V \times N$ enthält für die obigen Modelle etwa 425 Millionen (v, n)-Paare. Daher konnte die Glättungskraft nur näherungsweise bestimmt werden. Hierzu wurde eine einfache *Monte-Carlo-Methode* benutzt: es wurden 1000 zufällige Paare aus dem $V \times N$-Raum generiert und der Prozentanteil der Paare mit positiver Wahrscheinlichkeit wurde als die näherungsweise bestimmte Glättungskraft des Modells angesehen. Abbildung 4.14 zeigt die Glättungskraft für die obigen Modelle. Auf der x-Achse wird wieder die Anzahl der Klassen aufgetragen. Startparameter hatten einen diesmal sehr kleinen Einfluss von 1%. Während die empirische Wahrscheinlichkeitsverteilung eine Glättungskraft von winzigen ca. 0,14% aufweist, hat eine Modell mit 50 Klassen und 50 Iterationen eine Glättungskraft von ca. 93%.

Für deutsche EM-basierte Klassifikationsmodelle wurde gemessen, dass die empirische Wahrscheinlichkeitsverteilung eine nahezu nicht vorhandene Glättungskraft besitzt (0.012%). Dies liegt daran, dass der $V \times N$-Raum in den deutschen Experimenten aus 2.5 Billionen Kombinationen besteht. Obwohl die *Baseline* ca. 10-mal kleiner ist als bei den englischen Modellen, weisen die deutschen Modelle für 25 Klassen eine gute Glättungskraft von ca. 32% auf. Abbildung 4.15 zeigt die vollständigen Resultate (für Modelle mit 100 Iterationen, Resultate gemittelt über die Startwerte).

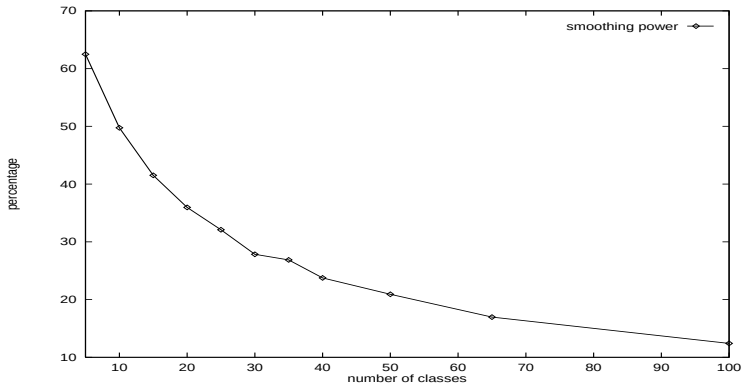


Abbildung 4.15: Glättungskraft der deutschen Modelle

Lexikon-Induktion: EM-basierte semantische Annotation

In diesem Abschnitt wird eine Methode zur automatischen Induktion von semantisch annotierten Subkategorisierungsrahmen aus freiem Text vorgestellt. Hierbei wird ein statistisches System benutzt, dass Korpus-Häufigkeiten von Paaren lexikalischer Köpfe in Prädikat-Argument-Relation estimiert (Carroll und Rooth 1998). Es wird sich zeigen, dass die Induktion der Annotationswahrscheinlichkeiten auf diesen Häufigkeiten basiert, sowie auf den Wahrscheinlichkeitsverteilungen, die ein fixiertes EM-basiertes Klassifikationsmodell

$$p_{\text{cluster}}(c),\ p_{\text{cluster}}(v|c),\ p_{\text{cluster}}(n|c), \qquad c \in C,\ v \in V,\ n \in N$$

über Verb-Nomen-Paare liefert.

Um die Wahrscheinlichkeiten der Klassen eines fixierten EM-basierten Klassifikationsmodells zu estimieren, die in der Subjekt-Position eines fixierten intransitiven Verbs vorkommen, wird das folgende Modell benutzt.

Annotationsmodell für intransitive Verben:

$$\begin{aligned} p(n) &= \sum_{c \in C} p(c,n) \\ &= \sum_{c \in C} p(c) \cdot p(n|c) \\ &= \sum_{c \in C} p(c) \cdot p_{\text{cluster}}(n|c). \end{aligned}$$

Hierbei bezeichnet $n \in N$ ein Nomen, das als lexikalischer Kopf in der Subjektposition eines gegebenen intransitiven Verbs auftaucht und $p(n)$ die entsprechende Wahrscheinlichkeit. Die Wahrscheinlichkeit $p(c)$ kann dann als die gesuchte Wahrscheinlichkeit interpretiert werden, dass die Klasse c (des gegebenen EM-basierten Klassifikationsmodells) in der

Subjekt-Position des intransitiven Verbs auftaucht. Die Benutzung von

$$p(n|c) \;=\; p_{\text{cluster}}(n|c)$$

bedeutet einerseits (rein technisch), dass die Wahrscheinlichkeiten $p(n|c)$ nicht als freie Parameter des Annotationsmodell angesehen werden, was technisch von Vorteil ist, weil die Zahl der zu estimierenden Parameter dann klein ist. Andererseits kann gerade mit Hilfe dieser Identifikation eine Brücke hin zu einem gegebenen Klassifikationsmodell geschlagen werden, um so methodisch dessen gesammelte Informationen für die Annotationsaufgabe auszunutzen.

Bezeichnet $f_v(n)$ die Häufigkeit eines Nomens n in einem *Korpus von gesammelten Subjekten zu dem fixierten intransitiven Verb* v, so liefert der EM-Algorithmus die folgenden einfachen Re-Estimierungsformeln.

EM-basierte Annotationswahrscheinlichkeiten für die Subjektklassen eines intransitiven Verbs: Nach jedem Re-Estimierungsschritt des EM-Algorithmus gilt für die re-estimierten Parameter $\hat{p}(c)$, unter Benutzung der gegenwärtigen Parameter $p(c)$, dass:

$$\hat{p}(c) \;=\; \frac{\sum_{n\in N} f_v(n)\cdot p(c|n)}{\sum_{n\in N} f_v(n)} \qquad (c \in C)\ .$$

Hierbei ist:

$$p(c|n) \;=\; \frac{p(c,n)}{p(n)} \;=\; \frac{p(c)\cdot p_{\text{cluster}}(n|c)}{p(n)}\ .$$

Natürlich sind die Annotationsmodelle und Trainingsverfahren für intransitive Verben relativ einfach auf transitive Verben erweiterbar. Um die Wahrscheinlichkeiten der Klassenpaare zu estimieren, die in der Subjekt- und Objektposition eines fixierten transitiven Verbs vorkommen, wird das folgende Modell benutzt:

Annotationswahrscheinlichkeiten für transitive Verben:

$$\begin{aligned} p(n,a) &= \sum_{c_n,c_a\in C} p(c_n,c_a,n,a) \\ &= \sum_{c_n,c_a\in C} p(c_n,c_a)\cdot p(n,a|c_n,c_a) \\ &= \sum_{c_n,c_a\in C} p(c_n,c_a)\cdot p_{\text{cluster}}(n|c_n)\cdot p_{\text{cluster}}(a|c_a)\ . \end{aligned}$$

Hierbei bezeichnet $(n,a) \in N \times N$ ein Nomenpaar, in dem das Nomen n als lexikalischer Kopf in der Subjektposition, und gleichzeitig das Nomen a in der Objektposition eines gegebenen intransitiven Verbs auftaucht. Die entsprechende Wahrscheinlichkeit dieses linguistischen Ereignisses wird mit $p(n,a)$ bezeichnet. Die Wahrscheinlichkeit $p(c_n,c_a)$ kann

dann als die gesuchte Wahrscheinlichkeit interpretiert werden, mit der die Klassen aus $(c_n, c_a) \in C \times C$ jeweils in Subjekt- und Objektposition des intransitiven Verbs vorkommen. Auch hier gilt, dass die Benutzung von

$$p(n, a|c_n, c_a) \;=\; p_{\text{cluster}}(n|c_n) \cdot p_{\text{cluster}}(n|c_a)$$

rein technisch gesehen, die Anzahl der freien Parameter des Annotationsmodells möglichst klein hält, andererseits die gesammelten Informationen eines semantischen Klassenmodells methodisch ausnutzt. Bezeichnet $f_v(n, a)$ die Häufigkeit des Nomenpaars (n, a) in einem Korpus von gesammelten Subjekten und Objekten eines fixierten intransitiven Verbs v, so liefert der EM-Algorithmus die folgenden einfachen Re-Estimierungsformeln.

EM-basierte Annotationswahrscheinlichkeiten für die Subjekt- und Objektklassen eines intransitiven Verbs: Nach jedem Re-Estimierungsschritt des EM-Algorithmus gilt für die re-estimierten Parameter $\hat{p}(c_n, c_a)$, unter Benutzung der gegenwärtigen Parameter $p(c_n, c_a)$, dass:

$$\hat{p}(c) \;=\; \frac{\sum_{n,a \in N} f_v(n, a) \cdot p(c_n, c_a|n, a)}{\sum_{n,a \in N} f_v(n, a)} \qquad ((c_n, c_a) \in C \times C) \;.$$

Hierbei gilt:

$$p(c_n, c_a|n, a) \;=\; \frac{p(c_n, c_a, n, a)}{p(n, a)} \;=\; \frac{p(c_n, c_a) \cdot p_{\text{cluster}}(n|c_n) \cdot p_{\text{cluster}}(a|c_a)}{p(n, a)} \;.$$

Bemerkungen:

1. Es ist interessant, dass die Annotationsmodelle auch für Verben v benutzt werden können, für die

   $$v \notin V$$

   gilt. D.h. ein Verb muss nicht im Trainingskorpus eines EM-basierten Klassifikationsmodells liegen, um den Verbrahmen mit Klassen dieses EM-basierten Klassifikationsmodells zu annotieren zu können. Dies ist so, weil in den Annotationsmodellen bewusst darauf verzichtet wurde, die Parameter

   $$p_{\text{cluster}}(v|c)$$

   einzubauen.

2. Für ein fixiertes transitives Verben wird mit Wahrscheinlichkeit $p(c_n, c_a)$ die Subjektposition mit c_n und die Objektposition mit c_a annotiert. Die Wahrscheinlichkeit

$p(c_n, c_a)$ ist also die Ereigniswahrscheinlichkeit von zwei *gleichzeitig eintretenden linguistischen Ereignissen*. Folglich ist

$$p(c_n) = \sum_{c_a \in C} p(c_n, c_a)$$

die Wahrscheinlichkeit, dass die Subjektposition mit Klasse c_n annotiert wird, und

$$p(c_a) = \sum_{c_n \in C} p(c_n, c_a)$$

die Wahrscheinlichkeit, dass die Objektposition mit Klasse c_a annotiert wird.

3. Die Implementierungen zum EM-basierten Aufbau eines semantisch annotierten Lexikons stehen lokal zur Verfügung (IMS, Universität Stuttgart, Prescher (1999)).

4. Es ist theoretisch interessant, dass die Lexikon-Induktion, wie die Induktion der EM-basierten Klassifikationsmodelle, mit einfachen probabilistischen Grammatiken vorgenommen werden kann (Prescher 2000).

5. Es ist interessant, dass die hier vorgestellten Annotationsmodelle nicht voraussetzen, dass die Wahrscheinlichkeitsverteilungen

$$p_{\text{cluster}}(c),\ p_{\text{cluster}}(v|c),\ p_{\text{cluster}}(n|c), \qquad c \in C,\ v \in V,\ n \in N$$

zu einem EM-basierten Klassifikationsmodell gehören. Die Annotationsmodelle könnten daher auch mit statistischen Versionen (Abney und Light (1999), Ciaramita und Johnson (2000)) alternativer semantischer Taxonomien, wie WordNet oder GermaNet, trainiert werden. Dieses Experiment ist geplant und soll insbesondere mit Taxonomien aus eng begrenzten Domänen arbeiten.

blush 5	0.982975	*snarl* 5	0.962094
constance	3	mandeville	2
christina	3	jinkwa	2
willie	2.99737	man	1.99859
ronni	2	scott	1.99761
claudia	2	omalley	1.99755
gabriel	2	shamlou	1
maggie	2	angalo	1
bathsheba	2	corbett	1
sarah	2	southgate	1
girl	1.9977	ace	1

Abbildung 4.16: Lexikoneinträge *blush*, *snarl*

increase 17	0.923698		
number	134.147	proportion	23.8699
demand	30.7322	size	22.8108
pressure	30.5844	rate	20.9593
temperature	25.9691	level	20.7651
cost	23.9431	price	17.9996

Abbildung 4.17: Annotation des Verbs *increase* mit der Klasse *der skalaren Änderung*

Experimente zur Lexikon-Induktion

In einem ersten Experiment mit englischen Daten wurde ein Modell mit 35 Klassen benutzt. Die wahrscheinlichsten Syntaxbäume, die ein statistischer Parser (Carroll und Rooth 1998) für den *British National Corpus* lieferte, wurden dazu benutzt, um die Häufigkeiten für intransitive Verben und ihre Subjekte zu bestimmen, sowie die Häufigkeiten für intransitive Verben und ihre lexikalischen Köpfe in Subjekt- und Objektposition zu extrahieren. Die 500 häufigsten Verben wurden dazu ausgewählt, semantisch annotiert zu werden.

Abbildung 4.16 zeigt die Lexikoneinträge für zwei Verben v, deren wahrscheinlichste Annotation die Klasse 5 ist. Diese Klasse wurde früher als *kommunikative Aktion* beschrieben. Ferner werden in der Abbildung die *estimierten Häufigkeiten*

$$fv(n) \cdot p(c|n)$$

für diejenigen 10 Nomen angegeben, für die diese estimierte Häufigkeit am höchsten ist.

Abbildung 4.17 zeigt ähnliche Lexikoneinträge für das intransitive Verb *increase* und die Klasse 17, die vorher als Klasse *skalarer Änderung* beschrieben wurde.

0.977992	decrease	0.560727	drop
0.948099	double	0.476524	grow
0.923698	increase	0.42842	vary
0.908378	decline	0.365586	improve
0.877338	rise	0.365374	climb
0.876083	soar	0.292716	flow
0.803479	fall	0.280183	cut
0.672409	slow	0.238182	mount
0.583314	diminish		

Abbildung 4.18: Steigerungsverben

increase (8, 17)	0.3097650
development - pressure	2.3055
fat - risk	2.11807
communication - awareness	2.04227
supplementation - concentration	1.98918
increase - number	1.80559

Abbildung 4.19: Transitives *increase* mit estimierten Frequenzen für das gemeinsame Auftreten der lexikalischen Köpfe

Abbildung 4.18 zeigt diejenigen intransitiven Verben, die die Klasse 17 als wahrscheinlichste Annotation haben. Intuitiv sind diese Verben semantisch kohärent. Wenn sie mit den 48 Top-Klassen von Levin (1993) verglichen werden, findet man mit Ausnahme der letzten drei Verben in der Liste in Abbildung 4.18, die absteigend nach der Annotationswahrscheinlichkeit sortiert ist, eine Übereinstimmung mit ihrer Klasse "verbs of changes of state".

Abbildung 4.19 zeigt die wahrscheinlichste Klassenpaar-Annotation für *increase* als transitives Verb. Ferner werden die estimierten Frequenzen

$$f_c(n,a) = f(n,a) \cdot p(c_n, c_a | n, a)$$

für das gemeinsame Auftreten der lexikalischen Köpfe gezeigt. Es ist bemerkenswert, dass die Klasse 17 auch für die intransitven Verben der skalaren Änderung die wahrscheinlichste Annotation ist.

Weitere Experimente wurden für deutsche Modelle mit 35 und 50 Klassen gemacht. Die nötigen Daten wurden für dieses Experiment aus den wahrscheinlichsten Syntaxbäumen extrahiert, die ein statistischer lexikalisierter Parser (Beil et al. 1998) für einen kleinen Korpus deutscher Verbletzt-Sätze (ca. 4,1 Millionen Wörter) lieferte.

bekanntgeben 14	0.999999	(make public)
Sprecher	4	(spokesman)
Polizei	3	(police)
Bundesamt	3	(Federal Agency)
Bürgermeister	2	(mayor)
Vorstands-Chef	2	(Chairman of the board)
Geschäftsleitung	2	(manager)
Vorstand	2	(board of management)
Unternehmen	1.99996	(company)
Wetteramt	1	(meteorological office)
Volksbank	1	(cooperative bank)

Abbildung 4.20: Intransitiver Lexikoneintrag: *bekanntgeben* (make public)

steigen 26	0.67273	(rise)
Zahl	23.333	(number)
Preis	15.895	(price)
Arbeitslosigkeit	10.8788	(unemployment)
Lohn	9.72965	(wage)
Nachfrage	6.83619	(demand)
Zins	6.80322	(interest)
Auflage	5.22654	(print run)
Beitrag	4.22577	(contribution)
Produktion	4.21641	(output)
Grundstückspreis	4	(price of a piece of land)

Abbildung 4.21: Intransitiver Lexikoneintrag: *steigen* (rise)

Abbildung 4.20 zeigt die Subjekte des intransitiven Verbs *bekanntgeben* (make public). Die Nomen werden (fast ausschlich) mit Klasse 14 annotiert. Diese Klase wurde früher als *Regierungsstelle, öffentliche Gewalt* beschrieben. Die Zahlen in den Spalten zeigen die estimierten Häufigkeiten

Abbildung 4.21 zeigt die Subjekte des intransitiven Verbs *steigen* (rise), welches mit Wahrscheinlichkeit 0.67273 mit Klasse 26 annotiert wurde, die ebenfalls als Klasse der *skalaren Änderung* interpretiert werden kann.

Ähnlich wie in den englischen Experimenten wurde beobachtet, dass sich auch die deutschen Verben der Klasse der skalaren Änderung in einem gewissen Sinn gleichverhalten. Abbildung 4.22 zeigt zehn intransitive Verben, die Klasse 14 aus einem 50-Klassen-Modell (die zur Klasse 26 in einem 35-Klassen-Modell korrespondiert) als wahrscheinlichste An-

0.741467	ansteigen	(go up)
0.720221	steigen	(rise)
0.693922	absinken	(sink)
0.656021	sinken	(go down)
0.438486	schrumpfen	(shrink)
0.375039	zurückgehen	(decrease)
0.316081	anwachsen	(increase)
0.215156	stagnieren	(stagnate)
0.160317	wachsen	(grow)
0.154633	hinzukommen	(be added)

Abbildung 4.22: Deutsche Steigerungsverben

senken (14, 26)	0.450352	(lower)
Bundesbank - Leitzins	5.81457	(Federal bank - base rate)
Bundesbank - Zins	2.97838	(Federal bank - interest)
Supermarkt - Preis	1	(super market - price)
Sommergeschäft - Verlust	1	(summer business - loss)
Bundesbank - Diskontsatz	0.99999	(Federal bank - minimum lending rate)
senken (14, 14)	0.147857	
Bundesbank - Lombardsatz	0.999973	(Federal bank - rate on loanes on security)
Strafandrohung - Abtreibungsquote	0.96842	(threat of punishment - abortion rate)
Strafandrohung - Abtreibungszahl	0.96842	(threat of punishment - number of abortions)
Fachhandel - Lagerkost	0.878333	(stores - storage charges)
Harmonisierung - Sozialniveau	0.764319	(harmonization - social level)

Abbildung 4.23: Transitiver Lexikoneintrag für *senken* (lower)

notation ihrer Subjektposition nehmen. Vergleicht Auf der Basis der wahrscheinlichsten Klassenannotation könnten diese Verben also als *Steigerungsverben* beschrieben werden. Vergleicht man diese maschinell erfolgte Klassifikation mit der linguistischen Klassifikation von Schuhmacher (1986), wird mit Ausnahme der Verben *anwachsen* (increase) und *stagnieren* (stagnate), die von Schuhmacher nicht untersucht werden, eine Übereinstimmung mit seiner Klasse von "einfachen Änderungsverben" gefunden.

Abbildung 4.23 zeigt ein Beispiel eines transitiven Lexikoneintrags, *senken* (lower). Klasse 14 wurde bereits als Klasse von *Regierungsstellen, öffentliche Gewalt* interpretiert, währen Klasse 36 die Klasse der *skalaren Änderung* ist.

Abbildung 4.24 zeigt das transitive Verb *dauern* (last/go on), für das Klassenpaar $(0, 10)$ eine sehr hohe Annotationswahrscheinlichkeit von 0,957095 besitzt. Klasse 0 kann als Klasse von *Projekten/Aktionen* interpretiert werden, während Klasse 10 eine *Zeit*-Bedeutung hat.

dauern $(0, 10)$	0.957095	(last/go on)
Entwirrung - Zeit	2	(disentanglement - time)
Bürgerfragestunde - Stunde	2	(question time - hour)
Prozess - Jahr	2	(trail - year)
Schreckenszeit - Jahr	1	(scaring time - year)
Ratenzahlung - Jahr	1	(buy in installments - year)

Abbildung 4.24: Transitiver Lexikoneintrag für *dauern* (last/to go on)

Diskussion

In diesem Abschnitt wurde ein Verfahren vorgestellt, um Subkategorisierungsrahmen mit lexikalischen Köpfen auf strukturierte Lexikoneinträge abzubilden. Dieses Verfahren ist wissenschaftlich interessant, praktisch einsetzbar und flexibel, denn:

1. Die Algorithmen und ihre Implementierungen sind effizient genug, um einen Korpus von ca. 100 Millionen Wörtern in ein Lexikon zu übertragen.

2. Das Modell und der Induktionsalgorithmus haben eine wohlfundierte Grundlage in der Theorie parametrisierter Wahrscheinlichkeitsverteilungen und ihrer Estimierung.

3. Die hergeleiteten lexikalischen Einträge sind linguistisch interpretierbar.

4. Aufgrund ihrer einfachen Gestalt sind die hergeleiteten Lexika vermutlich ihrerseits in der Parsing-Technologie, beispielsweise für lexikalisierte Grammatiken, aber auch in vielen anderen Anwendungen wiederverwendbar.

5. Die Methode ist für jede natürliche Sprache einsetzbar, für die es Texte genügender Grösse, sowie eine maschinelles Morphologiesystem und einen robusten Parser gibt, der fähig ist, Subkategorisierungsrahmen mit lexikalischen Köpfen zu extrahieren.

4.4 Identifikation von Kollokationen

In diesem Abschnitt werden Ergebnisse vorgestellt, die aus einer gemeinsamer Arbeit mit Ulrich Heid stammen und in Prescher und Heid (2000) veröffentlicht wurden.

In Abschnitt 4.3 wurde ein maschinelles Verfahren zum Aufbau semantisch annotierter Lexika vorgestellt. Die Erstellung eines EM-basierten Klassifikationsmodells von Verb-Nomen-Paaren unter Benutzung eines grossen Textkorpus und eines statistischen Parsers (Beil et al. 1999) spielte in diesem Verfahren eine Schlüsselrolle, da die Modellklassen in einem zweiten Schritt dazu benutzt werden konnten, um die Subjekt- und Objektpositionen intransitiver und transitiver Verben semantisch zu annotieren. Eines der zentralen Merkmale des EM-basierten Klassifikationsmodells, welche die Klassifikation semantischer Klassen überhaupt möglich macht, ist, dass die Verben und Nomen innerhalb der Klassen des Modells frei miteinander kombinieren sollen.

Ziel dieses Abschnitts ist, ein maschinelles Verfahren zur Identifikation von Verb-Nomen-Kollokationen in Textkorpora vorzustellen, dass erneut als Schlüsselschritt ein EM-basiertes Klassifikationsmodell benutzt. Da sich Kollokationen gerade dadurch auszeichen, dass (i) die beteiligten Lexeme nur selten auch mit anderen Partnern kombinieren und (ii) die Semantik der Kollokation sich nicht aus der Semantik der beteiligten Lexeme erschliessen lässt, kann das hier vorgestellte, neue Verfahren als Ergänzung des in Abschnitt 4.3 vorgestellten Verfahrens zum Aufbau eines semantisch annotierten Wörterbuchs aufgefasst werden.

Es gibt eine Reihe von interessanten statistischen Ansätzen zur maschinellen Identifikation von Kollokationen. Einige werden in Kilgarriff (1996) und Lezius (1999) vorgestellt. Das bisher erfolgreichste und daher am häufigsten angewandte statistische Verfahren zur Identifikation von Kollokationen stammt von Dunning (1993). Dunning zeigt empirisch, dass statistische Testverfahren für *seltene linguistische Ereignisse* bessere Ergebnisse erzielen, wenn man deren zugrundeliegenden Wahrscheinlichkeitsverteilungen anstatt mit der Normalverteilung durch eine Binomialverteilung modelliert. Eine neue Idee von Johnson (2001b) basiert ebenfalls auf der mathematischen Test-Theorie und verallgemeinert in gewisser Weise die Idee von Dunning. Es wird ebenfalls empirisch gezeigt, dass eine vorsichtig ausgeführte Konstruktion von *Konfidenzintervallen für Binomialverteilungen* in der Lage ist, die Parameter der Wahrscheinlichkeitsverteilungen seltener linguistischer Ereignisse ebenfalls sehr gut zu estimieren.

Das hier vorgestellten Verfahren ist ein Resultat einer Frage von Kamp (1999). Dieser hatte offenbar die Intuition, dass mit den zweidimensionalen EM-basierten Klassifikationsmodellen, die darauf basieren, dass Verben und Nomen innerhalb der Klassen frei miteinander

kombinieren, es eventuell auch möglich sein sollte, Verb-Nomen-Kombinationen zu identifizieren, bei denen die freie Kombinierbarkeit ganz offensichtlich gerade nicht gegeben ist.

Identifikation von Verb-Nomen-Kollokationen unter Benutzung von EM-basierten Klassifikationsmodellen für Verb-Nomen-Datentypen

Es wird von dem folgenden (irgendwo eingeführten) Begriff der *Kollokation* ausgegangen: während das Nomen die *semantisch autonome Basis der Kollokation* ist, ist das Verb ein *synsemantischer Kollokator*, d.h. dass das als Kollokator benutzte Verb in der Kollokation eine spezifische, gegebenenfalls abgeschwächt-schematisierte Bedeutung hat (siehe Helbig (1984) für den Fall von Funktionsverbgefügen), wenigstens aber eine Bedeutung hat, die nicht ganz der erwarteten kompositionell ableitbaren entspricht. Kollokationen folgen weitgehend den syntaktischen Konstruktionsprinzipien für Verb-Komplement-Konstruktionen. Einer der spezifischen Aspekte von Kollokationen (und der Grund dafür, dass man Kollokationen eigens lernen muss) ist die Tatsache, dass die Kombination der Lexeme von Basis und Kollokator nicht vorhergesagt werden kann. Vorhersagbarkeit ist im Rahmen der syntaktischen Bedingungen bei freier Kombinierbarkeit gegeben, oder, im Sinne von Subregularitäten, bei Selektionsklassen von Basen (anhand Sorten beschreibbare Kookkurrenz, (Heid 1994)).

Im folgenden soll versucht werden, diese beiden Annahmen (in der Kollokation ist die Wahl des Kollokators nicht frei; die Bedeutung der Kollokatoren weicht von der 'regulären' Bedeutung derselben Lexeme in freien, 'trivialen' Kombinationen ab) für ein korpusbasiertes Experiment zur Kollokationsextraktion nutzbar zu machen. Heid (1998) stellt die Motivation von korpusbasierten Ansätzen zur Kollokationsfindung vor. In den vorangegangenen Abschnitten wurden die EM-basierten Klassifikationsmodelle vorgestellt und gezeigt, dass diese Modelle erfolgreich zur Disambiguierung und Klassifizierung linguistischer Daten eingesetzt werden können. Während für die erste Aufgabe die Klassen nur als eine generelle Hintergrundquelle für die semantisch geglättete Wahrscheinlichkeitsverteilung für ungesehene oder seltene Datenpaare dienen, sind sie für die zweite Aufgabe, der Aufdeckung/Klassifizierung der in den Daten steckenden lexikalischen Semantik sehr relevant.

Das Training eines EM-basierten Klassifikationsverfahrens für Verb-Nomen-Datentypen ist ein mathematisch wohldefiniertes statistisches Inferenzverfahren, dessen wesentliche Grundannahme ist, dass die Verben einer jeden Klasse frei mit den Nomina dieser Klasse kombinieren können. Gerade diese Annahme trifft aber auf Verb-Nomen-Kollokationen nicht zu, da sie als eine Kombination von einem Verb und einem Argument angesehen werden könnten, die aus irgendeinem Grund statistisch präferiert wird. Somit kann vermutet

werden, dass ein gegebenes EM-basiertes Klassifikationsmodell den Kollokationen, die ja die grundlegende Modellannahme verletzen, eine eher kleine Wahrscheinlichkeit zuweisen wird, um sie den Paaren zu geben, die sich modellkonform verhalten, d.h. die in ihrer jeweiligen Klasse frei mit anderen Partnern kookkurrieren. Im folgenden soll diese Idee formalisiert werden, um ein Bewertungs-Kriterium dafür zu erhalten, ob ein gegebenes Nomen-Verb-Paar als Kollokation anzusehen ist.

Ein EM-basiertes Klassifikationsmodell $p_{\text{cluster}}(.)$ macht für jede seiner Klassen $c \in C$ die folgende *probabilistische Unabhängigkeitsannahme*:

$$p_{\text{cluster}}(v, n|c) = p_{\text{cluster}}(v|c) \cdot p_{\text{cluster}}(n|c) \ .$$

Da Kollokationen diese Unabhängigkeitsannahme verletzen, wird angenommen, dass eine Kollokation (v, n) von ihrer empirischen Wahrscheinlichkeit

$$\tilde{p}(v, n) = \frac{f(v, n)}{|f|}$$

während des Trainings des EM-basierten Klassifikationsmodells einen gewissen Teil an modellkonforme Verb-Nomen-Paare verlieren wird. In sehr vereinfachter Überlegung wird deshalb erwartet, dass die Differenz von empirischer Wahrscheinlichkeit und Modell-Wahrscheinlichkeit:

$$\tilde{p}(v, n) - p_{\text{cluster}}(v, n)$$

für Kollokationen positiv ist (weil diese sich während des Trainings des EM-basierten Klassifikationsmodells nicht modellkonform verhalten), und für "triviale" Verb-Nomen-Paare negativ ist.

$\hat{p} - p_{\text{cluster}}$	**Verb-Nomen-Kombination**
7.25709e-05	begehen_VPA.na:NP.AKK Straftat
2.38116e-06	spendieren_VPA.nad:NP.Akk Bier
7.41134e-05	treffen_VPP.n:NP.Nom Entscheidung
2.3708e-06	zerstören_VPP.n:NP.Nom Vegetation
2.73341e-05	schweigen_VPA.n:NP.Nom Waffe
4.77741e-06	spenden_VPA.nad:NP.Nom Franz

Die Tabelle zeigt einige Beispiele bei denen diese Annahme offensichtlich erfüllt ist. Die benutzte Notation ist weitgehend mit der in Abschnitt 4.3 benutzten Notation identisch. Ergänzend kommt hinzu, dass *.VPP* im Gegensatz zu *.VPA* ein passives Verb bezeichnet.

Von einer Liste von Verb-Nomen-Paaren, die absteigend nach den Werten der Wahrscheinlichkeitsdifferenz $\tilde{p}(v, n) - p_{\text{cluster}}(v, n)$ sortiert ist, kann daher erwartet werden, dass die Verb-Nomen-Kollokationen am Kopf dieser Liste zu finden sind.

Experiment

In einem Experiment mit deutschen Daten wurden 318 086 Datentypen von Verb-Nomen- oder Adjektiv-Nomen-Paaren aus den wahrscheinlichsten Synatxbäumen eines 4,1 Mio. Wort-Korpus deutscher Verb-Letzt-Sätze extrahiert (Beil et al. 1998). Diese Daten dienten als Eingabe des in Abschnitt 4.3 vorgestellten zweidimensionalen EM-basierten Klassifikationsverfahrens. Für die Verb-Nomen-Paare aus dem Trainingskorpus eines EM-basierten Klassifikationsmodells mit 35 Klassen wurde die Bewertung $\tilde{p}(v,n) - p_{\text{cluster}}(v,n)$ berechnet und absteigend sortiert. Ein erster Blick auf die Ergebnisse legt nahe, dass in der Tat, die Kombinationen, die besonders viel Wahrscheinlichkeitsmasse abgegeben haben, intuitiv als signifikate Kollokationen angesehen werden: Beispiele sind:

- *Rolle + spielen, Problem + lösen, Erfolg + haben, Gebrauch + machen, Ziel + erreichen, Anspruch + haben, Spass + machen* etc.

Am Ende der sortierten Liste treten primär kompositionell analysierbare, der jeweiligen Selektionsklasse entsprechende Wortgruppen auf:

- *der Polizei vorliegen, Garten + betreten, der Stadt guttun, Arbeitsplatz + aufgeben, Gegner + sehen.*

Evaluierung

Ziel der Untersuchungen ist es, Anhaltspunkte für die Grenzen zwischen Kollokation und Selektion zu bekommen; dass im allgemeinen Fall keine ganz klare Grenze gezogen werden kann, ist weithin akzeptiert. Trotzdem können Lexikographen relativ gut zwischen kollokatorischen und nicht-kollokatorischen Kombinationen unterscheiden. Benutzen sie ein Werkzeug, das ihnen aus Corpora extrahierte Kollokationskandidaten vorschlagen soll, so ist natürlich ihr Interesse, möglichst viel kollokatorische Kombinationen und wenig Triviales zu erhalten. Die im Experiment erhaltene Liste von Verb-Nomen-Paaren (v,n), absteigend nach den Werten $\tilde{p}(v,n) - p_{\text{cluster}}(v,n)$ sortiert, stellt ein solches Werkzeug dar und kann daher nach diesem Kriterium qualitativ evaluiert werden. In diesem Fall bedeutet es, dass unter den Kombinationen, die ganz oben in der Liste stehen, möglichst viele Kollokationen zu finden sein sollten.

In Abbildung 4.25 werden die 20 aktiven Verben und Objekte mit den höchsten $\tilde{p} - p_{\text{cluster}}$-Werten gezeigt (Top-20). Als Objekte wurden sowohl Akkusativ- als auch Dativobjekte zugelassen. In Abbildung 4.26 werden zum Vergleich die Top-20 passiven Verben und

0.000344138	haben_VPA.nai:NP.Akk nichts
0.000287041	spielen_VPA.nap:NP.Akk Rolle
0.00021156	lösen_VPA.na:NP.Akk Problem
0.000201252	haben_VPA.nd:NP.Dat Erfolg
0.000165406	machen_VPA.nd:NP.Dat Gebrauch
0.000143906	erreichen_VPA.na:NP.Akk Ziel
0.000126935	haben_VPA.nap:NP.Akk Anspruch
0.000124071	geben_VPA.nap:NP.Akk Problem
0.00011495	machen_VPA.nad:NP.Akk Spass
0.000113718	haben_VPA.nap:NP.Akk Einfluss
0.00011103	geben_VPA.nai:NP.Akk nichts
0.000109662	kosten_VPA.na:NP.Akk Mark
0.000107914	wissen_VPA.na:NP.Akk nichts
0.000106176	haben_VPA.nd:NP.Dat Bestand
0.000104177	leisten_VPA.nap:NP.Akk Beitrag
0.000102838	haben_VPA.nd:NP.Dat Vorrang
9.98574e-05	haben_VPA.nap:NP.Akk Interesse
9.32472e-05	treffen_VPA.na:NP.Akk Entscheidung
8.23121e-05	nachkommen_VPA.nd:NP.Dat Verpflichtung
8.17559e-05	nehmen_VPA.nd:NP.Dat Schaden

Abbildung 4.25: Aktive Verben und Objekte

0.000144875	lösen_VPP.n:NP.Nom Problem
0.00011534	erfüllen_VPP.n:NP.Nom Forderung
7.94366e-05	lösen_VPP.np:NP.Nom Problem
7.68178e-05	erreichen_VPP.n:NP.Nom Ziel
7.41134e-05	treffen_VPP.n:NP.Nom Entscheidung
6.23074e-05	treffen_VPP.np:NP.Nom Entscheidung
5.60728e-05	erzielen_VPP.np:NP.Nom Einigung
5.53494e-05	besetzen_VPP.n:NP.Nom Stelle
5.16534e-05	ändern_VPP.n:NP.Nom Grundgesetz
4.68707e-05	ausüben_VPP.np:NP.Nom Druck
4.67291e-05	erheben_VPP.n:NP.Nom Anklage
4.46287e-05	tragen_VPP.nd:NP.Nom Rechnung
4.37018e-05	einstellen_VPP.n:NP.Nom Verfahren
4.14156e-05	erfüllen_VPP.n:NP.Nom Bedingung
3.94744e-05	schaffen_VPP.np:NP.Nom Arbeitsplatz
3.94025e-05	schaffen_VPP.n:NP.Nom Voraussetzung
3.74812e-05	erfüllen_VPP.np:NP.Nom Forderung
3.56396e-05	bieten_VPP.nd:NP.Nom Einhalt
3.53789e-05	schaffen_VPP.n:NP.Nom Arbeitsplatz
3.48676e-05	machen_VPP.np:NP.Nom Fehler

Abbildung 4.26: Passive Verben und Subjekte

0.000214584	mitteilen_VPA.n:NP.Nom Polizei
0.000129361	bestehen_VPA.n:NP.Nom Gefahr
0.000125135	passieren_VPA.n:NP.Nom etwas
8.14615e-05	ändern_VPA.npr:NP.Nom etwas
8.1289e-05	steigen_VPA.n:NP.Nom Zahl
8.05926e-05	fehlen_VPA.n:NP.Nom Geld
7.79403e-05	mitteilen_VPA.n:NP.Nom Sprecher
6.65567e-05	fallen_VPA.n:NP.Nom Entscheidung
6.41839e-05	ändern_VPA.nr:NP.Nom etwas
6.40753e-05	ausgehen_VPA.np:NP.Nom Gefahr
6.10802e-05	eintreffen_VPA.n:NP.Nom Polizei
6.09915e-05	kommen_VPA.np:NP.Nom Rede
5.93555e-05	kann_VPA.np:NP.Nom Rede
5.72543e-05	vergehen_VPA.nd:NP.Nom sehen
5.70229e-05	sagen_VPA.n:NP.Nom Sprecher
5.66463e-05	melden_VPA.n:NP.Nom Nachrichtenagentur
5.62966e-05	bestehen_VPA.n:NP.Nom Interesse
5.57909e-05	zeigen_VPA.n:NP.Nom Beispiel
5.24866e-05	aufgehen_VPA.n:NP.Nom Rechnung
5.24282e-05	wiederholen_VPA.nr:NP.Nom Geschichte

Abbildung 4.27: Aktive Verben und Subjekte

Subjekte angegeben (Subjekt-Objekt-Alternation). Schliesslich werden in Abbildung 4.27 noch die Top-20 aktiven Verben und Subjekte gezeigt.

In der qualitativen Evaluierung wurden zum Beispiel folgende erwartete Nomen-Verb-Kollokationen gefunden (sortiert nach Nomen):

Arbeit aufnehmen_VPA.na:NP.Akk
Arbeit finden_VPA.na:NP.Akk
Arbeit leisten_VPA.na:NP.Akk
Arbeit machen_VPA.na:NP.Akk
Arbeit verrichten_VPA.na:NP.Akk
Verantwortung tragen_VPA.na:NP.Akk
Verantwortung übernehmen_VPA.na:NP.Akk
Weg einschlagen_VPA.na:NP.Akk
Weg gehen_VPA.na:NP.Akk

Sowie (sortiert nach Verben):

annehmen_VPA.na:NP.Akk Angebot
annehmen_VPA.na:NP.Akk Ausmass
annehmen_VPA.na:NP.Akk Form
annehmen_VPA.nd:NP.Dat Gestalt
abgeben_VPA.na:NP.Akk Erklärung
abgeben_VPA.na:NP.Akk Schuss
abgeben_VPA.na:NP.Akk Stellungnahme
abgeben_VPA.na:NP.Akk Stimme

Die folgende Liste zeigt Kombinationen mit *Quantoren*, die eventuell als Kollokationen anerkannt werden könnten, die aber nicht der Standard-Definition für Kollokationen genügen:

nichts anhaben_VPA.nad:NP.Akk
nichts bringen_VPA.na:NP.Akk
nichts nützen_VPA.na:NP.Akk

Abgesehen von Parsing-Fehlern:

- Kollokationen mit Präpositionalphrasen
- *erblassen_VPA.nd:NP.Dat Neid / vor*

- *geraten_VPA.nd:NP.Dat Kontrolle / ausser*
- *zurückbleiben_VPA.nd:NP.Dat Erwartung / hinter*

• Subjekt/Verb falsch erkannt

- *fliegen_VPA.na:NP.Akk Fetzen*
- *platzen_VPA.nad:NP.Akk Kragen*

wurden zum Beispiel folgende triviale Kombinationen fälschlich als Kollokationen identifiziert:

- *betragen_VPA.na:NP.Akk Prozent*
- *brauchen_VPA.na:NP.Akk Zeit, Geld, Hilfe*
- *bringen_VPA.na:NP.Akk Entlastung, Erfolg, Ergebnis, Lösung*
- *darstellen_VPA.nap:NP.Akk Gefahr*
- *verspinnen_VPA.nd:NP.Dat Schafwolle*

Neben der qualitativen Evaluierung wurden die Ergebnisse mit der anerkannt besten Methode zur Identifizierung von Kollokationen verglichen.

Die Abbildung 4.28 zeigt die Ergebnisse eines Experiments, in der eine Liste von Top-400 Kollokationskandidaten nach der hier vorgestellten Methode erstellt wurde. Zum Vergleich wurde mit den in Dunning (1993) vorgestellten *Log-Likelihood-Scores* unter Benutzung derselben Eingabedaten eine zweite Top400-Liste von Kollokationskandidaten erstellt. Um die Akuratheit beider Methoden zu vergleichen, wurde eine dritte Liste erstellt, in welcher die beiden Top-400 Listen zusammengelegt und alphabetisch sortiert wurden. Dies ergab eine Liste von rund 500 Kollokationskandidaten. Ein Kollokationsexperte (U.H.) markierte dann die echten Kollokationen auf dieser Liste. Anschliessend wurde für jedes $1 \leq n \leq 400$ gemessen, wieviele Kollokationen auf der Top-n-Liste zu finden waren. Abbildung 4.28 entnimmt man, dass beide Methoden in dieser Evaluierung annähernd dieselbe Performanz aufweisen, wobei die EM-basierten Klassifikationsmodelle in kürzeren Top-Listen (bis zu 300 Kandidaten) sogar mehr echte Kollokationen finden, als die Methode von Dunning.

Zusammenfassung

Ausgehend von zwei zentralen Annahmen des in der Lexikographie verbreiteten Kollokationsbegriffs wurden Kollokationen aus Textcorpora extrahiert.

Als Resource wurde ein grosser deutscher Textkorpus, sowie ein statistischer Parser benötigt, mit dem es möglich ist, Subkategorisierungsrahmen zusammen mit lexikalischen

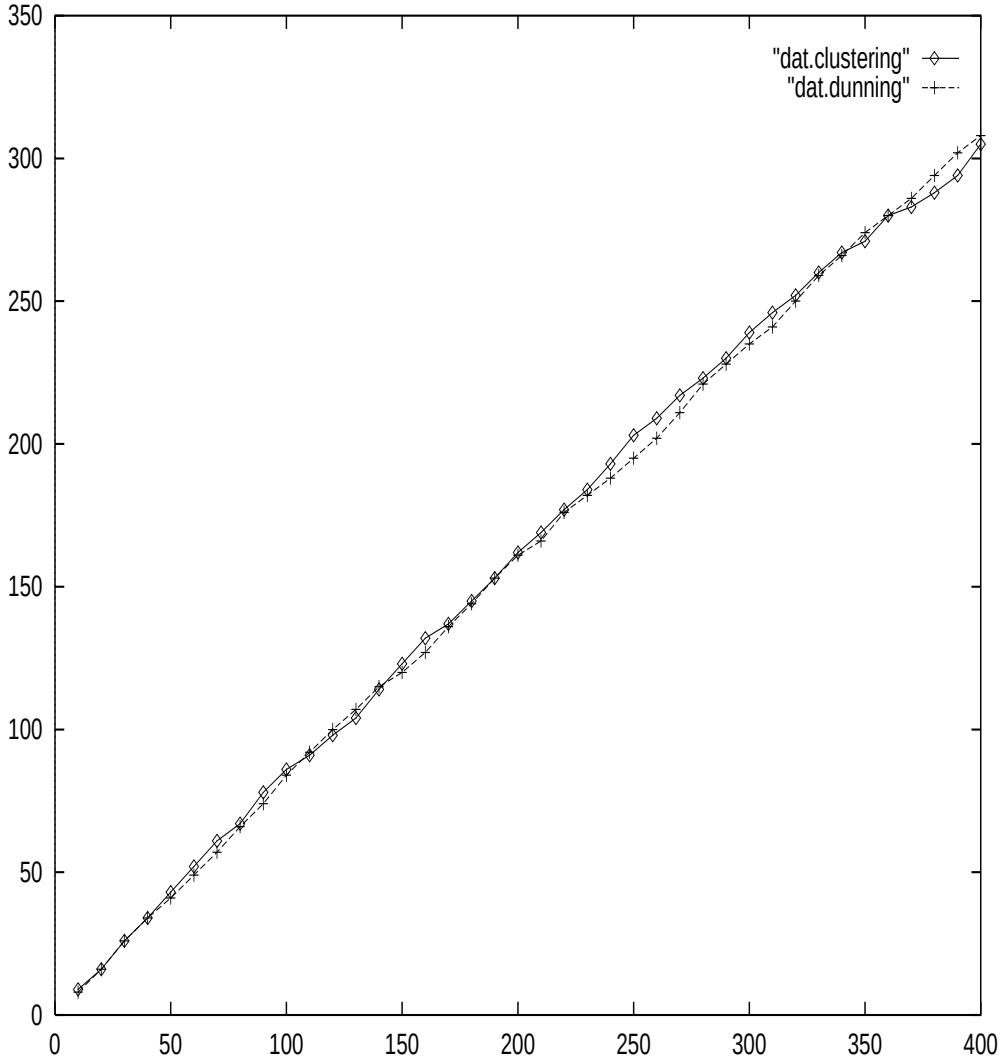


Abbildung 4.28: Akuratheit der Top-400-Liste im Vergleich mit der Akuratheit der Log-Likelihood-Scores (Dunning, 1993). Hierbei wird die Anzahl der echten Kollokationen (y-Achse) gegen die Anzahl der Kandidaten abgetragen (x-Achse)

Köpfen zu extrahieren. Dieses Datenmaterial wurde mit einem EM-basierten Klassifikationsverfahren semantisch klassifiziert. Da in jeder dieser Klassen die Nomen mit den Verben frei kombinieren können, und sich die Semantik einer Klasse aus der Semantik der darin vorkommenden Verben und Nomen erschliesst, sind Kollokationen nicht modellkonform.

Ein einfacher Vergleich der semantisch geglätteten Wahrscheinlichkeitsverteilung mit der empirischen Wahrscheinlichkeitsverteilung (Differenzbildung) war in der Lage, Kollokationen zu identifizieren. In einem Experiment wurde mit dem neuen Verfahren eine Top-400 Liste deutscher Kollokationen erstellt. Der Anteil nicht-kollokatorischer Kombinationen lag in dieser Liste bei nur ca. 25% und war damit vergleichbar mit den Ergebnissen, die mit der zur Zeit besten Methode zur Identifikation von Kollokationen erzielt wurden.

Obwohl die Resultate also sehr vielversprechend sind, muss in zukünftigen Experimenten sorgfältiger evaluiert werden, indem z.B. *sämtliche Kollokationen* in einem *grossem Korpus* oder zumindest in einer grossen Stichprobe dieses Korpus annotiert werden, um so auch messen zu können, wieviele der Kollokationen im Korpus durch eine Top-Liste abgedeckt werden (Evert et al. 2000). Ferner müssen Experimente mit besseren Vergleichen der empirischen Wahrscheinlichkeitsverteilung $\tilde{p}$ mit der semantisch geglätteten Wahrscheinlichkeitsverteilung p_{cluster} gefahren werden. Hier bieten sich anstelle der simplen Differenzbildung eine Reihe von alternativen Massen aus der Informationstheorie an, an erster Stelle die *(punktweise) relative Entropie.*

4.5 Lexikalische Disambiguierung

In diesem Abschnitt werden Ergebnisse vorgestellt, die sich aus einer gemeinsamen Arbeit mit Stefan Riezler und Mats Rooth ergaben und in Prescher et al. (2000) veröffentlicht wurden.

Die Disambiguierung lexikalischer Ambiguitäten wird in der maschinellen Sprachverarbeitung als eine schwierige Aufgabe betrachtet. Bei der Disambiguierung von Wortbedeutungen ist zum Beispiel die Aufgabe zu lösen, jedem Vorkommen eines ambigen Wortes eine Bedeutungsannotation zuzuordnen, die der Bedeutung dieses Wortes im jeweiligen Kontext entspricht. Eine Lösung dieses Problems könnte natürlich sehr hilfreich sein, um die Anzahl der möglichen semantischen Interpretationen von grösseren lexikalischen Einheiten sinnvoll zu beschränken. In der maschinellen Übersetzung gibt es ein ähnliches Problem: die Disambiguierung von Zielworten. Hier muss eine Entscheidung getroffen werden, welches aus einer Menge möglicher Zielworte die angemessenste Übersetzung eines Quellwortes ist. Eine Lösung dieses Disambiguierungsproblems wäre zum Beispiel direkt in jedem Übersetzungssystem einsetzbar, das in der Lage ist, eine Menge von Übersetzungsalternativen vorzuschlagen. Ein weiteres Problem betrifft zum Beispiel die Auflösung von *Attachment-Ambiguitäten* beim syntaktischen Parsing. Hier könnten die Entscheidungen, eine Präpositionalphrase an eine Verbphrase oder eher an eine Nomenphrase anzubinden, darauf basieren, wie die lexikalischen Ambiguitäten der jeweiligen Phrasenköpfe aufgelöst werden.

Auf alle diese Probleme wurden bereits statistische Ansätze mit grossem Erfolg angewandt. Der Vorteil von statistischen Methoden gegenüber symbolischen Methoden scheint zu sein, dass die statistischen Methoden selbst minimales linguistisches Wissen sehr effektiv ausnützen können. Leider stützen sich jedoch die besten statistischen Ansätze zur lexikalischen Disambiguierung auf komplexe Informationsquellen, wie "lemmas, inflected forms, parts of speech and arbitrary word classes [...] local and distant collocations, trigram sequences, and predicate argument association" (Yarowsky (1995), Seite 190) oder auf grosse Kontextfenster mit bis zu 1000 Nachbarworten (Schütze 1992). Unglücklicherweise stehen aber in vielen interessanten Anwendungen diese reichhaltigen Informationsquellen nicht zur Verfügung. In der maschinellen Übersetzung könnte es zum Beipsiel sehr erwünscht sein, die wahrscheinlichste Übersetzung von Verbargumenten allein unter Benutzung der Übersetzung des Verbes zu bestimmen, ohne auf die nur schwer zugreifbaren Informationen angewiesen zu sein, die in den Kontextfenstern alternativer Übersetzungen zu finden sind. Beim Parsing könnte es vorteilhaft sein, die Anbindung eines nominalen Kopfes auflösen zu können, indem lediglich die Information genutzt wird, die in der semantischen Rolle

des Verbs steckt, aber Informationen aus weiteren Prädikat-Argument-Relationen nicht genutzt werden.

Das Ziel dieses Abschnitts ist daher, eine Disambiguierungsmethode vorzustellen, die minimale, jedoch präzise Informationen benutzt. Es wird gezeigt werden, dass durch ein einfaches Nachschlagen in einem semantisch annotierten Lexikon (siehe 4.3) sehr gute Disambiguierungsresultate erzielbar sind. Jeder Eintrag eines solchen Lexikons besteht im wesentlichen aus einer Wahrscheinlichkeitsverteilung über semantische Klassen, mit denen die Subjekt- und Objektpositionen intransitiver und transitiver Verben annotiert werden. Jede dieser Wahrscheinlichkeitsverteilungen wird hierbei durch eine eigene Anwendung des EM-Algorithmus auf einen Korpus von Argumenten eines fixierten Verbs gewonnen. Die Disambiguierung möglicher Zielworte kann mit diesem Lexikon auf eine sehr naheliegende Weise vorgenommen werden, indem im Lexikon nach demjenigen Nomen (unter den möglichen Zielworten) gesucht wird, dessen Semantik am besten mit der Semantik des Zielverbs übereinstimmt. Diese einfache Methode wird auf einer grossen Anzahl von Übersetzungen eines natürlichen bilingualen Korpus evaluiert. Es wird sich zeigen, dass sie ähnlich gute Ergebnisse liefert, wie die von Dagan und Itai (1994), in der aber wesentlich mehr Selektionsinformationen benutzt werden.

Aufbau semantisch annotierter Lexika mit dem EM-basierten Klassifikationsverfahren für Verb-Argument-Datentypen

Für die in diesem Abschnitt vorgestellten Experimente wurde das in Abschnitt 4.1 vorgestellte EM-basierte Klassifikationsverfahren auf zwei-dimensionale Verb-Argument-Datentypen angewendet. Mit dem erhaltenen EM-basierten Klassifikationsmodell wurde anschliessend, wie in Abschnitt 4.3 beschrieben, ein semantisch annotiertes Lexikon aufgebaut.

Der Trainingskorpus für das EM-basierte Klassifikationsmodell bestand aus einem Korpus von 1 178 698 Datentoken (608 850 Datentypen) von Verb-Nomen-Paaren, die an den grammatischen Relationen intransitiver und transitiver Verben mit ihren Subjekt- und Objektpositionen beteiligt waren. Abbildung 4.29 zeigt eine induzierte Klasse aus einem englischen Model mit 35 Klassen. Klasse 19 kann als *lokative Aktion* interpretiert werden, da sie Nomen wie *area* und *world* und Verben wie *enter* und *cross* enthält.

Die Wahrscheinlichkeit, dass mit einer dieser Klassen die Objektposition eines fixierten transitiven Verbs annotiert wird, wurde mit einer weiteren Instanz des EM-Algorithmus bestimmt (siehe Abschnitt 4.3). In einem Experiment mit englischen Datentypen wurde der *British National Corpus* mit einem statistischen lexikalisierten Parser geparst, um anschliesssend für alle Sätze aus dem jeweils wahrscheinlichsten Syntaxbaum die Vor-

Class 19 PROB 0.0235		0.0250	0.0211	0.0125	0.0111	0.0106	0.0096	0.0085	0.0084	0.0081	0.0079	0.0076	0.0068	0.0067	0.0061	0.0059	0.0056	0.0053	0.0050	0.0049	0.0049	0.0048	0.0048	0.0047	0.0047	0.0046	0.0046	0.0045	0.0044	0.0043	0.0043
		room	area	world	meeting	range	school	service	building	road	street	market	course	doctor	place	part	mind	class	scene	path	group	work	gray	programme	line	life	garden	body	miles	system	period
0.0629	enter.aso:o	•	•	•		•	•	•	•	•	•	•		•	•	•	•	•	•	•	•	•	•	•		•	•	•		•	•
0.0386	cover.aso:o		•	•	•	•	•	•		•	•	•	•	•	•	•			•		•	•	•	•	•			•	•		•
0.0321	call.aso:o	•	•	•	•		•	•		•	•				•	•	•	•	•	•	•	•	•	•	•	•				•	•
0.0236	include.aso:o	•	•	•	•	•	•	•	•		•		•	•	•	•	•	•			•	•	•	•	•	•	•	•	•	•	•
0.0226	run.aso:o	•		•			•	•					•	•	•	•		•		•	•	•		•	•	•			•	•	
0.0214	attend.aso:o				•		•	•		•		•		•	•	•		•	•		•	•		•							
0.0173	cross.aso:o	•	•			•			•	•	•		•			•	•			•			•		•		•	•			
0.0136	dominate.aso:o	•	•	•	•			•	•	•	•	•			•			•	•			•	•			•		•		•	•
0.0132	have.aso:s	•		•	•	•	•	•	•	•	•	•	•		•	•	•	•	•	•		•	•	•	•	•	•	•	•	•	•
0.0126	attract.aso:s		•		•		•	•		•			•		•	•			•		•	•		•					•	•	•
0.0124	occupy.aso:o	•	•	•		•					•			•	•	•	•						•					•	•		
0.0115	include.aso:s	•	•		•	•	•	•	•	•		•	•	•	•	•	•	•	•	•		•	•		•	•	•	•		•	•
0.0113	contain.aso:s	•	•	•	•	•	•	•	•		•	•	•	•	•	•		•	•		•	•		•	•	•	•	•		•	•
0.0108	become.as:s	•	•	•	•	•	•		•	•	•	•		•	•	•	•	•		•	•	•	•	•	•	•	•	•		•	•
0.0099	form.aso:o		•	•				•		•		•						•			•	•	•		•			•		•	•
0.0086	collapse.as:s			•	•			•	•	•		•		•		•					•	•				•		•	•	•	
0.0085	create.aso:o		•	•		•		•			•	•		•	•		•	•	•	•	•	•	•	•	•	•	•	•		•	•
0.0082	provide.aso:s	•	•	•	•	•	•	•	•	•		•	•	•	•	•		•	•	•	•	•	•	•	•	•	•	•		•	•
0.0082	organize.aso:o	•			•	•		•					•			•		•			•	•		•		•		•			
0.0082	offer.aso:s	•	•	•	•	•	•	•	•	•	•	•	•	•	•			•		•	•	•		•	•	•	•	•		•	

Abbildung 4.29: Klasse 19: "locative Aktion"

kommenshäufigkeiten für die transitiven Verben mit ihren Objekten zu extrahieren. Diese Vorkommenshäufigkeiten wurden benutzt, um ein kleines semantisch annotiertes Lexikon mit 336 Einträgen zu bauen.

Abbildung 4.30 zeigt die obersten Teile der Lexikoneinträge für die zwei transitiven Verben *cross* und *mobilize*. Die Klasse 19 ist die wahrscheinlichste Klasse für die Objekte von *cross* (Wahrscheinlichkeit 0.692); die Objekte von *mobilize* gehören mit Wahrscheinlichkeit 0.386 zu Klasse 16, welches ebenfalls die am wahrscheinlichsten annotierte Objektklasse ist. Abbildung 4.30 zeigt für jedes Verb die zehn Nomen n mit der höchsten *klassenbasierten Häufigkeit*

$$f_c(n) = f(n) \cdot p(c|n) \ .$$

Hierbei ist $f(n)$ die Häufigkeit von n im Korpus aller gefundenen Objekte des gegebenen Verbs. Damit kann berechnet werden, dass in der Klasse 19 ca. 74-mal das Nomen *mind* als Objekt von *cross* gefunden wird, während das Nomen *force* in Klasse 6 das häufigste Objekt von *mobilize* sein wird.

Disambiguierung mit dem semantisch annotierten Lexikon

Im folgenden wird der einfache, aber natürliche Nachschlage-Mechansimus vorgestellt, der im folgenden als Disambiguierungsmethode eingesetzt werden soll. Abbildung 4.31 zeigt

cross.aso:o 19	0.692
mind	74.2
road	30.3
line	28.1
bridge	27.5
room	20.5
border	17.8
boundary	16.2
river	14.6
street	11.5
atlantic	9.9

mobilize.aso:o 6	0.386
force	2.00
people	1.95
army	1.46
sector	0.90
society	0.90
worker	0.90
member	0.88
company	0.86
majority	0.85
party	0.80

Abbildung 4.30: Klassenbasierte Häufigkeiten der Objekte der transitiven Verben *cross* und *mobilize*

(ID 160867) *Es gibt einige alte Passvorschriften, die besagen, dass man einen Pass haben muss, wenn man die* **Grenze** *überschreitet.* There are some old provisions regarding passports which state that people crossing the **{border/ frontier/ boundary/ limit/ periphery/ edge}** should have their passport on them.

(ID 201946) *Es gibt schliesslich keine Lösung ohne die Mobilisierung der bürgerlichen* **Gesellschaft** *und die Solidarität der Demokraten in der ganzen Welt.* There can be no solution, finally, unless civilian **{company/ society/ companionship/ party/ associate}** is mobilized and solidarity demonstrated by democrats throughout the world.

Abbildung 4.31: Beispiele für Zielwort-Ambiguitäten

zwei bilinguale Sätze aus dem in diesem Experiment benutzten Evaluierungskorpus. Die Quellwörter und ihre entsprechenden Zielwörter sind in **fetter Schrift** hervorgehoben. Die korrekte Übersetzung des Quellnomens (z.B. *Grenze*), wie sie von den tatsächlichen Übersetzern gewählt wurde, wird in der Abbildung nicht gezeigt, da sie durch eine Menge von alternativen Übersetzungen (z.B. { border, frontier, boundary, limit, periphery, edge }) ersetzt wurde, die ein kleines Wörterbuch Deutsch nach Englisch (siehe Abbildung 4.33) lieferte.

Das zu lösende Problem besteht darin, eine korrekte Übersetzung des Quellwortes zu finden, die nur minimale kontextuelole Information benutzt. In dem hier vorgestellten Ansatz fällt die Entscheidung für eines von mehren alternativen Zielnomen lediglich unter Berücksichtung des subkategorisierenden Zielverbs. Die Schlüsselidee ist, diese minimale Information maximal anzureichern, indem die präzise und komprimierte Information aus einem semantisch annotierten Lexikon hinzugefügt wird. Das Kriterium für die Wahl eines der alternativen Zielnomen ist daher, wie gut seine lexikalische und semantische Information mit der lexikalischen und semantischen Information des Zielverbs übereinstimmt. Dieses Kriterium kann durch ein simples Nachschlagen in einem semantisch annotierten Lexikon überprüft werden, wobei dasjenige Zielnomen gesucht wird, welches für ein gegebenes Zielverb die höchste klassenbasierte Häufigkeit aufweist. Formal wird dasjenige Zielnomen $\hat{n}$ (und diejenige Klasse $\hat{c}$) gesucht, für die:

$$f_{\hat{c}}(\hat{n}) = \max_{n\in,c\in C}\ f_c(n)\ .$$

Hierbei ist

$$f_c(n) = f(n) \cdot p(c|n)$$

die klassenbasierte Häufigkeit von n in dem Korpus der Argumente des fixierten Zielverbs, $p(c|n)$ ist die Wahrscheinlichkeit, dass das Nomen n mit der Klasse c annotiert wird, und $f(n) \geq 1$ ist die Häufigkeit von n in dem kombinierten Korpus von Objekten und Übersetzungsalternativen[4].

Im Beispiel mit der ID 160867 aus Abbildung 4.31 muss unter den angegebenen direkten Objekten des Verbs *cross* die Wahl getroffen werden. Der Lexikoneintrag von *cross* wird in Abbildung 4.30 gezeigt. Klasse 19 und das Nomen *border* erzielen eine höhere klassenbasierte Häufigkeit als jede andere Kombination aus einer Klasse und einem alternativen Zielnomen, wie zum Beispiel *boundary*. Ganz ähnlich hat im Beispiel ID 301946 das Paar, das aus dem Zielnomen *society* und Klasse 6 besteht die höchste klassenbasierte Häufigkeit unter allen Objekten von *mobilize*.

[4]Es gilt $p(\hat{c}) = \max_c p(c)$ zwar nicht in allen, aber in fast allen Fällen.

ambiguity	human baseline	clustering	ProbLex
2	73.5 %	79.0 %	88.3 %

Abbildung 4.32: Evaluierung mittels Pseudo-Disambiguierung von Nomen

Evaluierung

Die vorgestellte Disambiguierungsmethode wurde anhand einer Pseudo-Disambiguierungsaufgabe evaluiert, wie sie etwa in Abschnitt 4.3 zur Evaluierung von EM-basierten Klassifikationsmodellen benutzt wurde.

Es wurde ein Evaluierungskorpus von 298 (v, n, n') Tripeln konstruiert, wobei jedes (v, n) ein ursprünglich gesehenes, aber nicht im Training benutztes Verb-Nomen-Paar ist. Das Nomen n' wurde dem Paar (v, n) dann zufällig, aber gemäss der beobachteten Nomen-Häufigkeit hinzugefügt. Die *Präzision der vorgestellten Disambiguierungsmethode* wurde als Prozentanteil der Entscheidungen gemessen, in der diese sich für das nicht-zufällige Zielnomen entschied ($\hat{n} = n$).

Wie in Abbildung 4.32 gezeigt wird, erreichte die Disambiguierung mit einem semantisch annotierten Lexikon eine Präzision von 88% (ProbLex), was eine Verbesserung von 9% gegenüber der Verwendung des besten EM-basierten Klassifikationsmodells ist (clustering), sowie eine Verbesserung von 15% gegenüber einer vom Menschen vorgenommenen Disambiguierung (human baseline) [5].

Die Resultate der Pseudo-Disambiguierung sollten in einer weiteren Evaluierung auf einer grossen Anzahl von zufällig ausgewählten Sätzen aus einem bilingualen Korpus bestätigt werden. Hierzu wurde der aus den *satzalignierten* Debatten des europäischen Parlaments bestehende mlcc-Korpus (multilingual corpus for cooperation) ausgewählt, der ca. 9 Millionen Worte für Deutsch und Englisch (und weitere Sprachen) enthält.

Mit diesem Korpus wurde auf die folgende Weise ein *Gold-Standard-Evaluierungskorpus* konstruiert. Zunächst wurden *online-verfügbare* Wörterbücher dazu benutzt, um eine Liste von Wort-zu-Wort-Übersetzungen zu extrahieren, sodass jede englische Übersetzung wenigstens 2-mal in dem mlcc-Korpus vorkam. Das resultierende Wörterbuch wird in Ab-

[5] Ähnliche Resultate konnten mit einer einfacheren Methode erzielt werden, die eine erneute Anwendung des EM-Algorithmus vermeidet. Hier werden $\hat{n}$ (und $\hat{c}$) so gewählt, dass

$$f_{\hat{c}}(v, \hat{n}) = \max_{c,n} \left((f_{\text{cluster}}(v, n) + 1) \cdot p_{\text{cluster}}(c|v, n) \right) .$$

Diese Methode hat allerdings den Nachteil, dass die Verben im EM-basierten Klassifikationsmodell repräsentiert sein müssen.

Angriff	aggression, assault, offence, onset, onslaught, attack , charge, raid, whammy, inroad
Art	**form, type, way**, fashion, fit, kind, wise, manner, species, mode, sort, variety
Aufgabe	**abandonment, office, task**, exercise, lesson, giveup, job , problem, tax
Auswahl	eligibility, selection, choice, varity, assortment, extract, range, sample
Begriff	concept, item, notion, idea
Boden	**ground, land, soil**, floor, bottom
Einrichtung	**arrangement, institution**, constitution, establishment, feature, installation, construction, setup, adjustment, composition, organization
Erweiterung	**amplification, extension**, enhancement, expansion, dilatation, upgrading, add-on, increment
Fehler	**error, shortcoming**, blemish, blunder, bug, defect, demerit, failure, fault, flaw, mistake, trouble, slip, blooper, lapsus Genehmigung permission, approval, consent, acceptance, approbation, authorization
Geschichte	**history, story**, tale, saga, strip
Gesellschaft	**company, society**, companionship, party, associate
Grenze	**border, frontier**, boundary, **limit**, periphery, edge
Grund	**master, matter, reason**, base, cause, ground, bottom root
Karte	**card, map**, ticket, chart
Lage	site, situation, position, bearing, layer, tier
Mangel	deficiency, lack, privation, want, shortage, shortcoming, absence, dearth, demerit, desideratum, insufficiency, paucity, scarceness
Menge	**amount, deal, lot**, mass, multitude, plenty, quantity, quiverful, volume, abundance, aplenty, assemblage , crowd, batch, crop, heap, lashings, scores, set, loads, bulk
Prüfung	**examination, scrutiny, verification**, ordeal, test, trial, inspection, tryout, assay, canvass, check, inquiry, perusal, reconsideration, scruting
Schwierigkeit	**difficulty, trouble**, problem, severity, ardousness, heaviness
Seite	**page, party, side**, point, aspect
Sicherheit	**certainty, guarantee, safety**, immunity, security , collateral , doubtlessness, sureness, deposit
Stimme	**voice, vote**, tones
Termin	**date, deadline, meeting**, appointment, time, term
Verbindung	**association, contact, link**, chain, conjunction, connection, fusion, joint , compound, alliance, catenation, tie, union, bond, interface, liaison, touch, relation, incorporation
Verbot	ban, interdiction, prohibition, forbiddance
Verpflichtung	**commitment, obligation, undertaking**, duty, indebtedness , onus, debt, engagement, liability, bond
Vertrauen	**confidence, reliance, trust**, faith, assurance, dependence, private, secret
Wahl	**election, option**, choice , ballot, alternative, poll , list
Weg	**path, road, way**, alley, route, lane
Widerstand	resistance, opposition, drag
Zeichen	character, icon, sign, signal, symbol, mark, token, figure, omen
Ziel	**aim, destination, end**, designation, target, goal, object, objective, sightings, intention, prompt
Zusammenhang	coherence, context, contiguity, connection
Zustimmung	**agreement, approval, assent**, accordance, approbation, consent, affirmation, allowance, compliance, compliancy, acclamation

Abbildung 4.33: Wörtebücher aus Online-Recourcen

bildung 4.33 gezeigt.

Auf der Grundlage dieses Wörterbuchs wurden alle bilingualen Satzpaare aus dem Korpus extrahiert, die sowohl das Quellnomen, als auch das Zielnomen enthielten. Von Hand wurden dann aus den resultierenden ca 10 000 Sätzen diejenigen Satzpaare gefiltert, in denen ein Quellnomen aus diesem Wörterbuch in der Objektposition eines transitiven Verbs vorkam. Ferner wurde auch verlangt, dass das Zielobjekt im Wörterbuch vorkam, sowie in der Objektposition einer akzeptablen Übersetzung des Quellverbs auftauchte. In dem bilingualen Satzpaar wurde das deutsche Nomen n_g im Quellsatz, seine englische Übersetzung n_e (ein von professionellen Übersetzern ausgewähltes Zielnomen), sowie das englische Verb v_e annotiert. Für das in Abbildung 4.33 gezeigte Wörterbuch von 35 deutschen Quellworten resultierte diese halbautomatische Prozedur in einen Evaluierungskorpus von 1 350 Beispielen. Die durchschnittliche Ambiguitätsrate beträgt in diesem Evaluierungskorpus 8,63 Übersetzungen pro Beispiel. Ferner wurden aus dem Wörterbuch 25 Quellworte ausgewählt, sowie diejenigen Übersetzungen dieser Quellworte, die semantisch klar unterschieden werden konnten. Die Einträge, die zu diesem Lexikon gehören, werden in Abbildung 4.33 in **fetter Schrift** gezeigt. Dieses kleinere Wörterbuch führte zu einem Evaluierungskorpus von 814 Beispielen mit einer durchschnittlichen Ambiguitätsrate von 2,83 Übersetzungen. Die Wörterbücher und die entsprechenden Evaluierungskorpora sind auf dem Internet verfügbar[6].

Es kann angenommen werden, dass eine Evaluierung auf diesen Korpora eine realistische Simulation der schweren Aufgabe ist, eine zielsprachige Disambiguierung per maschineller Übersetzung vornehmen zu müssen, so wie sie in realen Situationen vorkommen wird: Die Übersetzungsalternativen wurden von online-verfügbaren Wörterbüchern ausgewählt, professionelle Übersetzer bestimmten die korrekten Übersetzungen in realen bilingualen Korpora, die Beispiele wurden zufällig ausgewählt ohne irgendwelche Beispiele zurückzuhalten, die durchschnittliche Ambiguitätsrate ist hoch und die alternativen Übersetzungen ähneln sich gelegentlich sehr. Im Gegensatz hierzu basieren die meisten anderen Evaluierungen auf nur zwei Alternativen mit klar unterschiedlichen Bedeutungen, die im allgemeinen eine hohe Korpus-Häufigkeit besitzen und den Experimentatoren auch noch in irgendeiner Weise als interessant erscheinen.

Abbildung 4.34 zeigt die Resultate der lexikalischen Disambiguierung mit semantisch annotierten Lexika (ProbLex) im Vergleich mit einfacheren Methoden. Die Tabellenzeilen zeigen die Resultate für die Evaluierungen auf den zwei Korpora mit den durchschnittlichen Ambiguitätsraten von 8,63 bzw. 2.83. Spalte 2 zeigt den Anteil der korrekten Übersetzungen, die eine Disambiguierung per Zufallsauswahl findet. Spalte 3 präsentiert eine andere

[6] `http://www.ims.uni-stuttgart.de/projekte/gramotron/`

ambiguity	random	major sense	empirical distrib.	clustering	ProbLex
8.63	14.2 %	31.9 %	P: 46.1 % E: 36.2 %	43.3 %	49.4 %
2.83	35.9 %	45.5 %	P: 60.8 % E: 49.4 %	61.5 %	68.2 %

Abbildung 4.34: Disambiguierungs-Resultate der EM-basierten Klassifikationsmodelle und der semantisch annotierten Lexika

Baseline: Disambiguierung mit der *Hauptbedeutung* (major sense), die ganz einfach darin besteht, das häufigste Zielnomen als Übersetzung des Quellnomens zu wählen. In Spalte 4 wird die empirische Wahrscheinlichkeitsverteilung der (v, n)-Paare aus dem *Britisch National Corpus* als Disambiguator benutzt (empirical distribution). Es ist interessant, dass diese Methode eine sehr gute *Präzision* (P = #korrekt / #korrekt + #inkorrekt) erzielt, aber eine sehr viel schlechtere *Akuratheit* (E = #korrekt / #korrekt + #inkorrekt + #weiss nicht) besitzt. Der Grund ist, dass einige (v, n)-Paare aus dem Evaluierungskorpus entweder nicht im *Britisch National Corpus* vorkamen, oder sich zumindest nicht von ihren Alternativen unterscheiden konnten, da diese dort mit der gleichen Häufigkeit auftauchten. Dieser Effekt konnte durch die EM-basierten Klassifikationsmodelle überwunden werden (Spalte 5, clustering). Hier wurde die *semantisch geglättete Wahrscheinlichkeitsverteilung* $p_{\text{cluster}}(v, n)$ der (v, n)-Paare benutzt, um zwischen alternativen Nomenpaaren zu entscheiden. Da die EM-basierten Klassifikationsmodelle den (v, n)-Paaren eine Wahrscheinlichkeit, d.h. einen kontinuierlichen Wert zwischen 0 und 1 zuweisen, der zudem aufgrund der hohen *Glättungskraft* der EM-basierten Klassifikationsmodelle meistens positiv ist, traten bei vergleichbarer Präzision sehr viel weniger "weiss nicht"-Fälle auf. Trotzdem ist auch die semantische geglättete Wahrscheinlichkeitsverteilung der EM-basierten Klassifikationsmodelle zu grob, wenn sie mit den Disambiguierungsergebnissen eines semantisch annotierten Lexikons verglichen wird. Hier ist auf beiden Korpora ein weiterer Zugewinn von 7% Präzision bei gleicher Akuratheit zu beobachten (Spalte 6). Es wird vermutet, dass dieser Zugewinn der Tatsache zugeordnet werden kann, dass in einem semantischen Lexikon die Kombination von Häufigkeitsinformationen für Nomen mit den Annotationswahrscheinlichkeiten für die Argumentklassen von Verben benutzt wird. Hierbei scheint es sehr wichtig zu sein, dass die Übersetzungsalternativen zu der Menge der beobachteten Verbobjekte hinzugefügt wurden, d.h. stets mit Häufigkeit gearbeitet wird, die nicht Null sind. Dies gibt dem Disambiguierungsmodell offenbar die Möglichkeit, zwischen ungesehenen Alternativen zu wählen, um so eine verbesserte Akuratheit zu erreichen. Ferner ist es möglich, dass die höhere Akuratheit nicht nur daraus resultiert, dass ProbLex die Nullen

source	target	correct	accept.
Seite	page side	76.9 %	76.9 %
Sicherheit	guarantee safety	93.8 %	93.0 %
Verbindung	connection link	58.8 %	93.8 %
Verpflichtung	commitment obligation	73.2 %	94.1 %
Ziel	objective target	72.5 %	85.5 %
overall precision		78 %	90 %

Abbildung 4.35: Akuratheit für das Auffinden von korrekten und akzeptablen Übersetzungen mit einem semantisch annotierten Lexikon

der empirischen Wahrscheinlichkeitsverteilung füllt, sondern auch besonders in den Fällen hilfreich ist, in denen die empirische Wahrscheinlichkeitsverteilung für zwei Alternativen denselben Wert hat.

Abbildung 4.35 zeigt die Disambiguierungsresultate mit einem semantisch annotierten Lexikon für 5 Wörter mit jeweils zwei Übersetzungen. Für dieses kleine Wörterbuch wurde ein Evaluierungskorpus von 219 Sätzen extrahiert. 200 dieser Sätze wurden mit der zusätzlichen Annotationen versehen, welches die akzeptablen Übersetzungen des Quellnomens sind. Als Resultat wurden 78% korrekte, sowie 90% akzeptable Übersetzungen gefunden.

Ferner wurde ein Experiment auf einer sehr kleinen Teilmenge von 100 Testsätzen mit einer durchschnittlichen Ambiguitätsrate von 8,6 Übersetzungsalternativen ausgeführt. Hier musste ein Mensch, dem nur das englische Verb, sowie die Menge der alternativen Zielnomen gegeben wurde, d.h. dieselbe Information, die auch die Modelle nutzten, zwischen den Übersetzungen wählen. Auf dieser Menge wurde eine menschliche Präzision von 39% gemessen.

Diskussion

Abbildung 4.36 zeigt einen Vergleich der Disambiguierungsergebnisse von ProbLex und den besten auf unannotierten Daten trainierten Disambiguierungs-Algorithmen. Spalte 2 (corpus size) zeigt die Anzahl der Beispiele in den Evaluierungskorpora der einzelnen Ansätze. Diese reicht von ca. 100 bis zu 37 000 Beispielen. ProbLex wurde auf Evaluierungskorpora mit 219, 814 und 1 340 Beispielen evaluiert.

disambiguation method	corpus size	ambiguity	random	precision	random (standardized)	precision (standardized)
ProbLex	1 340	8.63	14.2 %	49.4 %	53.4 %	79.7 %
	814	2.83	35.9 %	68.2 %	50.5 %	77.5 %
	219	2	50.0 %	78.0 %	50.0 %	78.0 %
Dagan, Itai 94	103	2.27	44.1 %	P: 91.4 % E: 62.1 %	50.0 %	P: 92.7 % E: 66.8 %
Resnik 97	88	3.51	28.5 %	44.3 %	50.0 %	63.8 %
SENSEVAL 00	2 756	9.17	10.9 %	P: 20-65 % E: 3-54 %	50.0 %	P: 60-87 % E: 33-83 %
Yarowsky 95	~ 37 000	2	50.0 %	96.5 %	50.0 %	96.5 %
Schütze 92	~ 3 000	2	50.0 %	92.0 %	50.0 %	92.0 %

Abbildung 4.36: Ein Vergleich von nicht überwachten lexikalischen Disambiguierungsmethoden

Spalte 3 (ambiguity) zeigt die durchschnittliche Anzahl von Bedeutungen bzw. Übersetzungen auf die die verschiedenen Disambiguierungsmethoden angewandt wurden. Diese variiert von 2 bis zu 9 Bedeutungen[7].

Spalte 4 (random) zeigt die Präzision der Zufalls-Disambiguierung, die für die einzelnen Experimente von ca. 11% bis 50% rangiert. Die Präzision der Verfahren wird in Spalte 5 (precision) angegeben. Um diese Resultate miteinander vergleichen zu können, war es erforderlich, die Ergebnisse auf eine einheitliche Ambiguitätsrate zu standardisieren. Dies wurde erreicht, indem der Wert

$$p^{1/\log_2 amb}$$

für die Präzision p und die Ambiguitätsrate amb berechnet wurde. Er entspricht der Präzision, die bei einer Ambiguitätsrate von 2 eintritt. Die Konsistenz dieser "Binarisierung" ist aus Spalte 6 ersichtlich, in der die Standardisierung der Zufalls-Disambiguierungen vorgenommen wurde und die tatsächlich für alle Disambiguierungsmethoden einen Wert von ca. 50% ergibt [8]. ProbLex erreicht eine standardisierte Präzision von ca. 79% auf allen Evaluierungskorpora. Den direktesten Vergleich erlaubt die Methode von Dagan und Itai (1994), die 91.4% Präzision (92.7% standardisiert) und 62.1% Akuratheit (66.8% standardisiert) erreicht. Diese Werte wurden allerdings auf einer relativ geringen Anzahl von 103 Testbeispielen für die Zielwortauswahl in einer Übersetzungsaufgabe von Hebräisch nach Englisch gemessen. Kompensiert man die hohe Präzision ihrer Methode mit der ge-

[7]Die Ambiguitätsrate von 2.27 aus dem Experiment von Dagan und Itai (1994) wurde berechnet, indem deren 3.27 alternativen Übersetzungen durch die durchschnittliche Anzahl von 1.44 korrekten Übersetzungen dividiert wurde. Die Ambiguitätsrate von 3.51 in dem Experiment von Resnik (1997) wurde mit der dort angegebenen Baseline von 28.5% Präzision für die Zufalls-Disambiguierung berechnet (100/28.5). Umgekehrt kann die Baseline für die Zufalls-Disambiguierung bei Dagan und Itai (1994) als 100/2.27 = 44.1 berechnet werden. Die Ambiguitätsrate für SENSEVAL wurde für die Nomen des englischen SENSEVAL - Evaluierungskorpus berechnet.

[8]Es ist klar, dass exakt 50% standardisierte Präzision erreicht werden, falls $precision \cdot amb = 100$ %, wie es bei der Präzision der Zufalls-Disambiguierung der Fall sein sollte

ringen Akuratheit ergeben sich wohl ähnliche Werte wie für ProbLex. Das Verfahren von Dagan und Itai (1994) basiert auf den Informationen, die aus möglichst allen in einem Satz vorkommenden grammatischen Relationen für verbale, nominale und adjektivische Prädikate gezogen werden, ohne allerdings klassenbasierte Informationen zu benutzen. Resnik (1997) präsentierte eine Disambiguierungsmethode, die auf einer Evaluierungsmenge von 88 Verb-Objekt-Datentoken 44.3 % Präzision (63.8 % standardisiert) lieferte. Sein Ansatz ist mit ProbLex insofern vergleichbar, als er ungefähr dieselbe Informiertheit des Disambiguators voraussetzt. Er benutzt ebenfalls klassenbasierte Masse, allerdings auf der Grundlage von WordNet-Klassen. Leider bestand die Aufgabe in seiner Evaluierung darin, WordNet-Klassen für die Verbobjekte auszuwählen, anstatt die Objekte selbst, sodass die Ergebnisse nicht direkt vergleichbar sind. Dasselbe trift für die SENSEVAL-Evaluierung zu (Kilgarriff und Rosenzweig 2000). Dort mussten Bedeutungen des HECTOR-Wörterbuchs disambiguiert werden. Die Präzision für die zehn nicht überwachten Systeme, die an der Konkurrenz teilnahmen, rangierte von 20-65% bei einer Akuratheit von 3-54%. Der SENSEVAL-Standard wird klar von den früheren Resultaten von Yarowsky (1995) (96.5 % Präzision) und Schütze (1992) (92 % Präzision) übertroffen. Aber auch hier ist ein Vergleich erneut etwas schwierig. Erstens wurden diese Ansätze anhand von Worten evaluiert, die von den Experimentatoren frei ausgewählt wurden und zwei sehr klar unterscheidbare Bedeutungen besassen, während ProbLex im Gegensatz dazu auf zufällig ausgewählten, realen Übersetzungen eines grossen bilingualen Korpus evaluiert wurde.

Ferner benutzen beide Ansätze einen riesigen Umfang an Informationen, z.B. sehr grosse Kontextfenster, oder sogar manuelle Intervention, z.B. initiale Bedeutungszuweisung (Yarowsky 1995). Solche Art von Information steht möglicherweise in IR-Anwendungen zur Verfügung, ist aber in Situationen, wie Parsing oder Übersetzung, sicher nicht vorhanden.

Zusammenfassung

Die hier präsentierte Disambiguierungsmethode ist mit voller Überlegung auf die Information begrenzt worden, die in einem semantisch annotierten Lexikon zur Verfügung steht. Diese Information bewies sich als akurat genug, um gute empirische Resultate in der Übersetzung zu zeigen. Nachdem ein semantisch annotiertes Lexikon aus einem unannotierten Korpus gebaut wurde, konnte die lexikalische Disambiguierung durch einen einfachen Nachschlagevorgang stattfinden. In der Zielwort-Disambiguierung wurde das häufigste Nomen gewählt, dessen Semantik am besten mit der Semantik des subkategorisierenden Verbs übereinstimmte. Die Methode wurde auf zufällig ausgewählten Beispielen aus realen bilingualen Korpora evaluiert, was eine realistische und schwere Aufgabe darstellt. Die gelernte

Lektion besagt, dass hybride Modelle, die Häufigkeits- und Klassen-Informationen benutzen, sowohl die reinen Häufigkeit-basierten Methoden, als auch die rein Klassen-basierten Methoden übertreffen.

4.6 Probabilistische Silbenmodelle

In diesem Abschnitt werden Ergebnisse vorgestellt, die aus gemeinsamer Arbeit mit Karin Müller und Bernd Möbius stammen und bereits in Müller et al. (2000b) und Müller et al. (2000a) veröffentlicht wurden oder in Müller (2001) vorgestellt werden.

In diesem Abschnitt werden probabilistische Silbenmodelle für Englisch vorgestellt, die mit dem in Abschnitt 4.1 eingeführten mehrdimensionalen EM-basierten Klassifikationsverfahren auf unannotierten drei- und fünfdimensionalen Daten trainiert wurden. Die erhaltenen Silbenmodelle wurden mit einem Pseudo-Disambiguierungsverfahren evaluiert. Hierbei konnte gezeigt werden, dass der *Onset* der variabelste Teil der Silbe ist. Die qualitative linguistische Evaluierung erbrachte, dass (ohne dass dies eigentlich vorab zu erwarten war) die Silbenmodelle tatsächliche bedeutungstragende Klassen besitzen.

Das EM-basierte Klassifikationsverfahren wurde für die Syntax entwickelt und dort für zweidimensionale Verb-Nomen-Datentypen angewandt (Rooth et al. 1999). Unglücklicherweise war dieser zweidimensionale Ansatz aber nicht auf Silben anwendbar, da angenommen werden muss, dass Silben wenigstens aus drei Dimensionen bestehen: *Onset, Nucleus* und *Coda*. Dies war die Motivation, das mehrdimensionale EM-basierte Klassifikationsverfahren zu entwickeln, das bereits in den vorangegangen Abschnitten dieses Kapitels vorgestellt wurde, in diesem Abschnitt aber tatsächlich das erste Mal in seiner vollen Mächtigkeit angewendet wird.

Experimente

Ein Korpus von Silben dient dem mehrdimensionalen EM-basierten Klassifikationsverfahren als Eingabe. Die englischen Daten kamen von dem *British National Corpus (BNC)*, einer Kollektion von geschriebener und gesprochener Sprache mit ca. 100 Millionen Worten. Die Silben wurden aufgesammelt, indem durch den Korpus gegangen wurde, um die Worte und ihre Silbifizierung in einem Aussprachewörterbuch nachzugeschlagen (Baayen et al. 1993)[9] und ihre Vorkommenshäufigkeiten zu zählen [10].

In dem ersten Experiment wurden dreidimensionale Modelle auf der Grundlage von Silben-Onset, -Nucleus und -Coda trainiert. Hierzu wurden 9 327 verschiedene deutsche Silben gesammelt. Die Anzahl der Klassen wurde von 1 bis 200 variiert. Abbildung 4.37 zeigt

[9] Dabei wurde eine leicht modifizierte Version des englischen Aussprachewörterbuchs verwendet, um stets nicht-leere Nuclei zu erhalten, z.B. wurde /idealism/ [aI][dI@][lIzm,] nach [aI][dI@][lI][z@m] übertragen (SAMPA-Transkription).

[10] Nachfolgende Experimente haben gezeigt, dass die Vorkommenshäufigkeiten eine wertvolle Information für die Klassifizierungsaufgabe darstellen (Müller et al. 2000b).

class 0	0.212	NOP[I]	0.282	I 0.999	NOP[I] 0.460
		t	0.107		n 0.121
		l	0.074		N 0.096
		d	0.071		z 0.079
		b	0.065		t 0.042
		s	0.060		ts 0.013
					f 0.012

Abbildung 4.37: Klasse #0 eines 3-dimensionalen englischen Modells mit 12 Klassen

einen Teil der Klasse #0 von einem drei-dimensionalen englischen Modell mit 12 Klassen. Die erste Spalte zeigt den Klassenindex und die Klassen-Wahrscheinlichkeit $p(0)$. Die wahrscheinlichsten Onsets und ihre Wahrscheinlichkeiten werden in absteigender Ordnung in Spalte 2 gezeigt, so wie Nucleus und Coda in der 3ten bzw. 4ten Spalte gezeigt werden. Leere Onsets und Codas wurden als "NOP[nucleus]" dargestellt. Klasse #0 enthält hochfrequente Funktionswörter *in, is, it, its* und Suffixe *-ing ,-ting, -ling*. Es ist interessant, dass diese Funktionswörter und Suffixe in den fünfdimensionalen Modellen in seperaten Klassen erscheinen (Klassen #1 und #3 in Abbildung 4.39).

In einem weiteren Experiment wurden fünfdimensionale Modelle induziert, die mit zwei zusätzlichen Parametern angereichert sind: die *Position der Silbe im Wort* und *Betonung*. Die Silbenposition hat vier Werte: einsilbig (ONE), initial (INI), medial (MED) und final (FIN). Betonung hat zwei Werte: betont (STR) und unbetont (USTR). Es wurden 16 595 verschiedene deutsche Silben gesammelt. Die Anzahl der Klassen rangierte erneut von 1 bis 200.

Pseudo-Disambiguation

Die dreidimensionalen Modelle wurden mittels Pseudo-Disambiguierung evaluiert, ähnlich wie es bereits in Abschnitt 4.2 beschrieben wurde, aber so verändert, dass das Verhalten des Modells bei Onset-, Nucleus,- und Coda-Ambiguität untersucht werden konnte.

Die erste Aufgabe besteht darin, zu entscheiden, welcher von zwei Onsets on and on' wahrscheinlicher im Kontext eines gegebenen Nucleus n und eines gegebenen Codas cod vorkommt. Zu diesem Zweck wurde ein Evaluierungskorpus von 3 000 Silben (on, n, cod) konstruiert, indem die Silben aus dem originalen Korpus gezogen wurden. Dann wurde zu jeder dieser Silben ein zufälliger Onset on' gewählt, sodass die Silbe (on', n, cod) weder im Trainings- noch im Evaluierungskorpus vorkam. Ferner wurde verlangt, dass alle Elemente on, n, cod, und on' Teil des Trainingskorpus waren. Die EM-basierten Klassi-

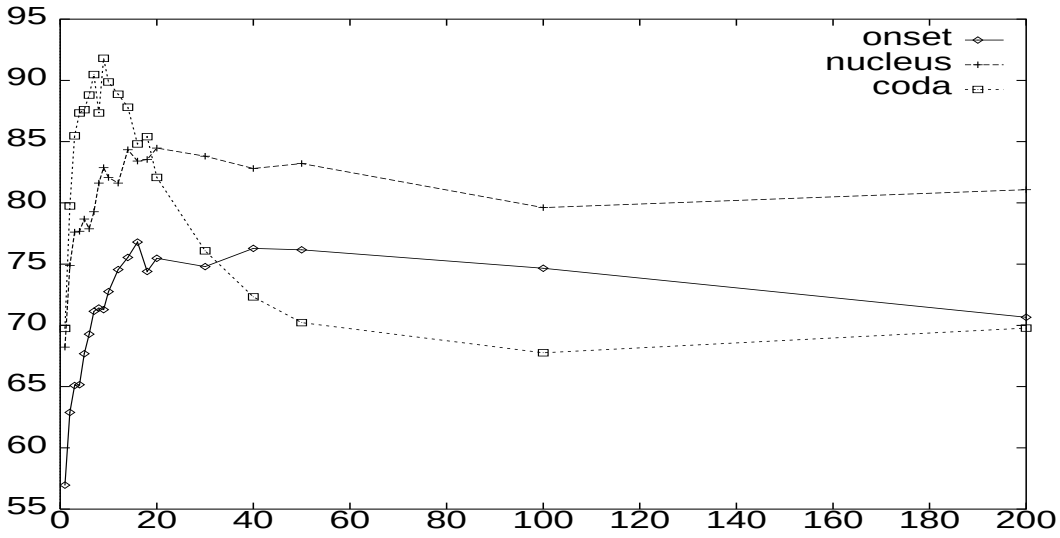


Abbildung 4.38: Evaluierung mittels Pseudo-Disamiguierung

fikationsmodelle wurden wie folgt parametrisiert: es gab bis zu zehn Startwerte für den EM-Algorithmus und bis zu 200 Klassen, was eine Folge von $10 \times 20 = 200$ zu evaluierende Modelle ergab. Akuratheit wurde auf dem Evaluierungskorpus als Prozentanteil der Modell-Entscheidungen $p(on, n, cod) \geq p(on', n, cod)$ gemessen. Zwei ähnliche Aufgaben wurden für Nucleus und Coda gestellt. Die Resultate, die mit den besten Startwerten erzielt wurden, werden in Abbildung 4.38 gezeigt. Modelle mit 12 Klassen zeigen die beste Akuratheit: 85-92% (Nucleus und Coda), sowie 77% (Onset). Dieses Resultat stimmt mit der Intuition überein:

(i) Der Onset ist der variabelste Teil der Silbe, da es einfacher ist, Mininmalpaare zu finden, die im Onset variieren,

(ii) es ist einfacher, Coda und Nucleus vorherzusagen, da ihre Wahl eingeschränkter ist.

Qualitative Evaluierung

Die folgende Evaluierung ist auf fünfdimensionale Silbenmodelle beschränkt, da die Qualität der Ausgabe steigt, je mehr Dimensionen hinzugefügt werden. Die Resultate können von mehreren Positionen beurteilt werden. Zum Beispiel könnte nachgeprüft werden, ob die Mehrzahl der Klassen von einem einzigen Nucleus dominiert wird (wie es bei den dreidimensionalen Modellen tatsächlich der Fall ist). Ein anderer interessanter Aspekt ist, ob es Silbenklassen gibt, die Teile von Inhaltswörtern, oder hochfrequente Funktionswörer

class 0 0.071	D 0.745 NOP[@] 0.166	@ 1	NOP[@] 0.877 m 0.0792	ONE 0.999	STR 1
class 1 0.049	NOP[I] 0.914 h 0.071 b 0.010	I 1	n 0.387 z 0.360 t 0.180 f 0.042 ts 0.02	ONE 0.916 INI 0.069	STR 1
class 3 0.040	t 0.206 s 0.106 d 0.104 NOP[I] 0.101 n 0.052	I 0.993	N 0.466 d 0.167 z 0.152 Nz 0.012	FIN 0.997	USTR 0.999
class 4 0.037	t 0.211 v 0.115 D 0.102 d 0.095 NOP[@] 0.072	@ 0.978 O: 0.009	r* 0.597 z 0.115 d 0.057 l 0.054 n 0.045	FIN 0.996 MED 0.003	USTR 0.999
class 10 0.028	S 0.257 m 0.227 d 0.063 t 0.059 NOP[@] 0.007	@ 0.926 I 0.031 I@ 0.015 E 0.005	n 0.388 nt 0.191 nz 0.088 l 0.066 nts 0.049 ns 0.048	FIN 0.999	USTR 0.999
class 14 0.026	m 0.116 p 0.108 k 0.090 g 0.088 t 0.080 pl 0.052 st 0.051	eI 0.426 A: 0.165 E 0.140 O: 0.110	t 0.162 s 0.131 n 0.088 d 0.079 k 0.079 nd 0.052 ts 0.037	ONE 0.696 FIN 0.276	STR 0.984
class 17 0.023	NOP[@] 0.973	@ 1	NOP[@] 0.325 r* 0.317	ONE 0.944 INI 0.050	STR 1

Abbildung 4.39: Klassen #0, #1, #3, #4, #10, #14, #17 eines 5-dimensionalen Modells für englische Silben

oder produktive Affixe repräsentieren.

Resultate. In 24 von 50 Silbenklassen wurde ein dominanter Nucleus pro Silbenklasse beobachtet. In allen diesen Fällen ist die Wahrscheinlichkeit dieses Nucleus höher als 99% und in 7 Klassen ist die Nucleus-Wahrscheinlichkeit 100%. Neben einigen Diphtongen dominieren nur die relativ seltenen Vokale /V/, /A:/ und /3:/ keine Klasse. Die Abbildung 4.39 zeigt alle Klassen, die im folgenden beschrieben werden.

Hochfrequente Funktionswörter werden von 10 Silbenklassen repräsentiert. Zum Beispiel werden die Klassen #0 und #17 von den Artikeln *the* bzw. *a* dominiert, und Klasse #1 enthält Funktionswörter, die den Kurzvokal /I/ enthalten, etwa *in, is, it, his, if, its.*

Produktive wortformende Suffixe werden in Klasse #3 (*-ing*), und flektierende Suffixe in Klasse #4 (*-er, -es, -ed*) gefunden. Klasse #10 ist besonders interessant, da es eine vergleichsweise grosse Anzahl von Suffixen repräsentiert, etwa *-tion, -ment, -al, -ant, -ent, -ence* und andere.

Die meisten Silbenklassen, nämlich 31 von 50 enthalten Silben, die mit einer hohen Wahrscheinlichkeit in initialer, medialer und finaler Position in offenen Wortklasssen im Lexikon

gefunden werden. Zum Beispiel enthält Klasse #14 hauptsächlich betonte Vokale /eI, A:, e:, O:/ und andere, die in einer Vielzahl von Silbenpositionen in Nomen, Adjektiven oder Verben vorkommen.

Zusammenfassung

Mit dem mehrdimensionalen EM-basierten Klassifikationsverfahren ist es möglich, phonologisch bedeutungsvolle Silbenklassen zu extrahieren.

Im Gegensatz zur Syntax (Rooth et al. 1999), wo semantische Relationen zwischen Verben und Objekten aufgedeckt wurden, konnte ein ähnliches Ergebnis für die Silbenmodelle nicht *a priori* erwartet werden. Silben-Konstituenten (oder Phone) repräsentieren ein sehr kleines Inventar von Einheiten, dass benutzt werden kann, um grössere bedeutungstragende Einheiten, Morpheme und Wörter, zu formen. Aber sie selbst tragen keine Bedeutung. Somit gibt es keinen Grund, warum gewisse Silbentypen signifikant häufiger als andere vorkommen sollten, ausser der Tatsache, dass gewisse Morpheme und Wörter mit einer höhere Häufigkeit als andere in einem gegebenen Korpus auftauchen. Daher ist es sehr interessant, dass gezeigt werden konnte, dass einige Silbenklassen dieser Überlegung zum Trotz interessante Eigenschaften zeigten: einige repräsentierten hochfrequente Funktionswörter und produktive Affixwe, während andere typischerweise in Inhaltswörtern gefunden wurden.

Eine Evaluierung mittels Pseudo-Disambiguierung zeigte schliesslich in Übereinstimmung mit der Intuition, dass der Onset der variabelste Teil der Silbe ist.

Kapitel 5

Stochastische Grammatiken

Gegenwärtig werden stochastische Grammatiken, die von prinzipiell grösserer Mächtigkeit als die probabilistischen kontextfreien Grammatiken sind, intensiv erforscht. Eine gemeinsame Eigenschaft der meisten dieser Modelle ist die Berücksichtigung von lexikalischen Köpfen in regelbasierten Wahrscheinlichkeitsmodellen, etwa in Collins (1997), Charniak (1997) und Ratnaparkhi (1997). Ein weiterer Aspekt dieser Ansätze ist, dass die Wahrscheinlichkeitsmodelle mit Baumbanken trainiert werden, d.h. mit Korpora von manuell annotierten Sätzen. In allen zitierten Ansätzen wird die Penn-Wall-Street-Journal-Baumbank (Marcus et al. 1993) benutzt, was ein Zeichen dafür ist, dass das Baumbank-Training aufwendig ist, da es die erhebliche manuelle Intervention erfordert, Korpora aus speziellen Domänen und speziellen Sprachen mit speziellen Syntaxanalysen zu versehen. Es wird allgemein angenommen, dass mit dem Training mit dem EM-Algorithmus (Dempster et al. 1977) nur schlechte Resultate erzielbar sind, wenn nicht zumindest partielle Annotationen verwendet werden. Experimentelle Resultate, die dieses "Allgemeinwissen" bestärken, werden z.B. in Elworthy (1994) und Pereira und Schabes (1992) für das EM-Training von Hidden-Markov-Modellen und probabilistischen kontextfreien Grammatiken präsentiert.

In diesem Kapitel sollen nicht diese verschiedenen Ansätze diskutiert werden, sondern es soll in diesem Kapitel gezeigt werden, dass der EM-Algorithmus, im Gegensatz zur weit verbreiteten Meinung, mit Erfolg eingesetzt werden kann, um akurate stochastische Grammatiken auf freiem Text, ohne Benutzung einer Baumbank, zu trainieren. Hierbei werden zwei alternative Ansätze vorgestellt: zum einen wird eine probabilistische lexikalisierte kontextfreie Grammatik vorgestellt, zum anderen eine stochastische lexikalisierte unifikationsbasierte Grammatik, die aber beide auf grossen Textkorpora trainiert werden. Beide Ansätze haben spezifische Stärken und Schwächen. Während probabilistische lexikalisierte kontextfreie Grammatiken mit dem Inside-Outside-Algorithmus, einer dy-

namischen Programmiervariante des EM-Algorithmus trainiert werden können, muss für das Training von stochastischen lexikalisierten unifikationsbasierten Grammatiken auf eine nicht-dynamische Instanz des EM-Algorithmus zurückgegriffen werden[1]. Dies hat zur Folge, dass die probabilistischen lexikalisierten kontextfreien Grammatiken effektiv trainiert und angewendet werden können, während es bei den stochastischen lexikalisierten unifikationsbasierten Grammatiken unerlässlich ist, die Anzahl der Analysen eines Satzes stark zu kontrollieren, d.h. extrem zu beschränken, damit der Formalismus anwendbar ist. Leider wird diese Eigenschaft nicht immer erfüllt sein. Der Nachteil der stochastischen lexikalisierten unifikationsbasierten Grammatiken wird aber möglicherweise dadurch ausgeglichen, dass sie in Experimenten, in denen die Anzahl der durchschnittlichen Analysen stark beschränkt war (≤ 25), sehr gute linguistische Ergebnisse zeigten. Insbesondere konnte gezeigt werden, dass die vorgenommene Lexikalisierung einen deutlich positiven Effekt hatte, was bei den probabilistischen lexikalisierten kontextfreien Grammatiken nicht der Fall war. Ferner konnte als wichtiger theoretischer Beitrag zur EM-Theorie formal begründet werden, dass für log-lineare Modelle (als Grundlage der stochastischen unifikationsbasierten Grammatiken) "optimale" *Startparameter* existieren. Dieses Ergebnis ist wichtig, weil der allgemeine EM-Algorithmus im Gegensatz hierzu anweist, möglichst viele Startparameter auszuprobieren. Nach bestem Wissen des Autors ist dies das erste Mal, dass eine theoretische Begründung für die Wahl bestimmter, fester Startparameter angegeben wird. Das restliche Kapitel gliedert sich wie folgt:

In Abschnitt 5.1 werden *probabilistische lexikalisierte kontextfreie Grammatiken* vorgestellt. Zunächst wird mit drei Beispielen (Briscoe und Carroll 1993) motiviert, dass diese Grammatiken einige natürlichsprachige Phänomene besser modellieren können, als gewöhnliche probabilistische kontextfreie Grammatiken. Dann werden *lexikalisierte Syntaxbäume* und *lexikalisierte kontextfreie Grammatiken* vorgestellt. Deren Grammatikregeln sind durch *Kopfmarkierungen* angereichert, um *lexikalische Köpfe* verwalten zu können. Das Wahrscheinlichkeitsmodell der lexikalisierten kontextfreien Grammatiken basiert auf *lexikalisch annotierten Grammatikregeln* und sogenannten *lexikalischen Entscheidungen*. Das Hauptziel dieses Abschnitt ist, anhand eines umfangreichen Beispiels vorzustellen, dass sich jede probabilistische lexikalisierte kontextfreie Grammatik durch eine *Grammatiktransformation* auf eine gewöhnliche probabilistische kontextfreie Grammatik zurückführen lässt (Carroll und Rooth 1998). Als Konsequenz ist der Inside-Outside-Algorithmus bzw. der EM-Algorithmus prinzipiell auch auf lexikalisierte kontextfreie Grammatiken anwendbar. Abschliessend wird vorgestellt, welche Nebenbedingun-

[1] Schmid (1999b) schlägt einen schnellen symbolischen Parsing- und Grammatik-Formalismus (YAP) für unifikationsbasierte Grammatiken vor, zusammen mit einer einfachen statistischen Disambiguierungskomponente.

gen die Wahrscheinlichkeitsparamter der lexikalisierten Grammatik erfüllen müssen, um die Standard-Nebenbedingung für kontexfreie Grammatiken nach Booth und Thompson (1973) zu erfüllen.

In Abschnitt 5.2 wird ein Experiment für Deutsch beschrieben, in dem auf einem Korpus von ca 450 000 Sätzen eine lexikalisierte kontextfreie Grammatik trainiert wird. Das Resultat dieses Experiments ist, dass lexikalisierte kontextfreie Grammatiken gegenüber unlexikalisierten Grammatiken zumindest gering verbesserte Evaluierungsergebnisse aufweisen werden. Hierfür wurden nicht die beim Baumbank-Trainingsverfahren üblichen "Exact-Match"-Evaluierungen vorgenommen, sondern sorgfältige linguistische Evaluierungen. Mit manuell erstellten, grossen Evaluierungskorpora wurde satzweise gemessen, wie gut Nominalphrasen und verschiedene Typen von Subkategoriseirungsrahmen von den Grammatiken erkannt werden können. Ein weiteres Ergebnis des Experiments ist, dass zu viele (mehr als zwei) Iterationen mit dem Inside-Outside-Algorithmus die Ergebnisse der linguistischen Evaluierung verschlechtern. Das gewöhnliche Kriterium, welches im allgemeinen benutzt wird, um Übertraining zu erkennen, nämlich die Messung der Korpuswahrscheinlichkeit eines *Held-Out-Korpus*, wies im Gegensatz zu diesen Ergebnissen an, sehr viel mehr Trainings-Iterationen vorzunehmen, sodass dieses Standard-Kriterium als unangemessen gelten kann.

In Abschnitt 5.3 wird ein neues stochastisches lexikalisiertes Modell für unifikationsbasierte Grammatiken vorgestellt, für welches der EM-Algorithmus gleich zweimal benutzt wird: zum einen wird eine Instanz des EM-Algorithmus für sogenannte *log-lineare Modelle* (Riezler 1999) benutzt, um das Modell auf freiem Text zu trainieren. Zum anderen basiert dieses Modell auf den vorab estimierten klassenbasierten Häufigkeiten eines EM-basierten Klassifikationsmodells für Verb-Argument-Paare (Prescher et al. 2000). In diesem Abschnitt werden *log-lineare Modelle* formal vorgestellt und in diesem Kontext die Begriffe: *Normalisierungskonstante*, *Parametervektor*, *Eigenschaften*, *Eigenschaftsvektor* und *Referenzverteilung* definiert. Ferner wird vorgestellt wie die Parameter so estimiert werden können, dass die empirische Wahrscheinlichkeitsverteilung der Trainingsdaten möglichst gut reflektiert wird. Das Hauptziel dieses Abschnitt ist es aber, theoretisch zu begründen, dass für log-lineare Modelle "optimale" *Startparameter* existieren. Dieses Ergebnis ist deshalb so wichtig, weil der allgemeine EM-Algorithmus im Gegensatz zu diesem Ergebnis anweist, möglichst viele Startparameter auszuprobieren.

In Abschnitt 5.4 wird ein Experiment für Deutsch präsentiert, in dem ein Korpus von ca. 36 000 Sätzen mit einer LFG-Grammatik geparst wird. Es werden folgende Schlüsse aus dem Experiment gezogen: (i) Die in Abschnitt 5.3 motivierten Startwerte bewähren sich in der Praxis hervorragend. (ii) Die Lexikalisierung mit klassenbasierten Häufigkeiten ver-

bessert die Präzision um ca. 10% (gegenüber unlexikalisierten Modellen). (iii) Der EM-Algorithmus verbessert die Estimierung der Wahrscheinlichkeitsparameter von unifikationsbasierten Grammatiken um ca. 10% Präzision (gegenüber einem überwachten Trainingsverfahren, das mit einer "Parsebank" arbeitet). Hierzu werden in diesem Abschnitt 190 überwiegend *nicht-lokale Basis-Eigenschaften* vorgestellt, sowie 45 weitere Eigenschaften, die die *Lexikalisierung des stochastischen Modells* ausmachen. Die experimentelle Evaluierung der Modelle wird mit dem üblichen "Exact-Match-Kriterium" (d.h. exakte Übereinstimmung der wahrscheinlichsten Syntaxanalyse mit der korrekten Syntaxanalyse) eine Präzision von 86% ergeben. Zur Messung dieser Ergebnisse wurde ein Evaluierungskorpus mit 550 manuell annotierten Sätzen mit durchschnittlich 5.4 Syntaxanlysen erstellt. Eine zweite Evaluierung, in der die Subkategorisierungsrahmen von Verben erkannt werden müssen, wird eine Präzision (Übereinstimmung der Subkategorisierungsrahmen des Hauptverbs in wahrscheinlichster und korrekter Syntaxanalyse) von 90% ergeben. Hier wurde ein Evaluierungskorpus mit 375 manuell disambiguierten Beispielen mit einer durchschnittlichen Ambiguität von 25 Syntaxanalysen erstellt.

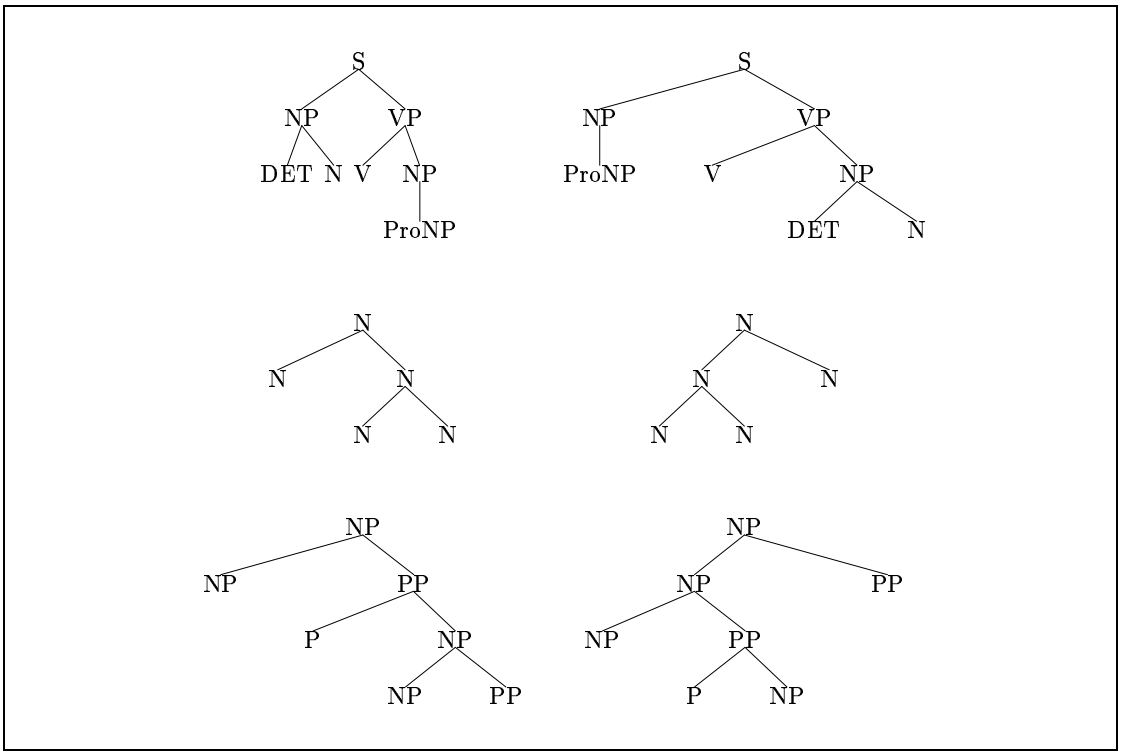


Abbildung 5.1: Strukturell ambige Syntaxbäume

5.1 Probabilistische lexikalisierte kontextfreie Grammatiken

Leider ist bekannt, dass das Wahrscheinlichkeitsmodell "gewöhnlicher" probabilistischer kontextfreier Grammatiken grundsätzlich nicht in der Lage ist, Syntaxbäume voneinander unterscheiden, die dieselben Grammatikregeln mit denselben Häufigkeiten benutzen.

In Abbildung 5.1 werden drei Beispiele für dieses schlechte Verhalten gezeigt, die sich auf drei verschiedene Phänomene im Englischen beziehen (Briscoe und Carroll 1993):

- Ein Pronomen wird im Englischen präferiert als Subjekt benutzt.
- Nomenkomposita: "toy coffee grinder" versus "cat food tin".
- Anbindung von Präpositionalphrasen:
  "the man in the park with the telescope".

Unabhängig von den Grammatikregelwahrscheinlichkeiten einer probabilistischen kontextfreien Grammatik werden in den drei Beispielen dem jeweils linken und rechten Syntaxbaum dieselbe Wahrscheinlichkeit zugeordnet, sodass nicht entschieden werden kann, ob die linke oder die rechte Analyse vorzuziehen ist. Leider hat dies zur Folge, dass die obenge-

nannten drei Phänomene im Englischen mit dem Standard-Wahrscheinlichkeitsmodells für kontextfreie Grammatiken nicht angemessen behandelt werden können. Glücklicherweise kann eine sogenannte *Lexikalisierung* der kontextfreien Grammatiken die angesprochenen Probleme beseitigen. Dies soll anhand des ersten Beispiels deutlich gemacht werden.

Eine Lexikalisierung der VP-Regel wird zu einer lexikalisierten Grammatik mit der folgenden Grammatikregel führen:

$$VP \rightarrow V' \; NP \; ,$$

was (informell) bedeutet, dass das Verb V der lexikalische Kopf der Verbphrase wird. Offensichtlich kommt dann in dem oberen, linken Syntaxbaum aus Abbildung 5.1 die *lexikalisch annotierte grammatische Relation*

$$VP[v]NP[pron]$$

vor, während im oberen, rechten Syntaxbaum

$$VP[v]NP[n]$$

vorliegt. Hierbei ist $VP[v]$ eine Verbphrase mit dem lexikalischen Kopf v, $NP[pron]$ eine Nominalphrase mit einem Pronomen *pron* und $NP[n]$ eine Nominalphrase mit einem Nomen n als lexikalischen Kopf. Da eine *probabilistische* lexikalisierte kontextfreie Grammatik diesen beiden grammatischen Relationen möglicherweise verschiedene Wahrscheinlichkeiten zuweisen wird, sind die betroffenen Syntaxbäume mit einer solchen Grammatik grundsätzlich disambiguierbar.

Das Ziel dieses Abschnitts ist es, *lexikalisierte kontextfreie Grammatiken* (relativ informell) vorzustellen und zu zeigen, dass es möglich ist, auch diese mit dem EM-Algorithmus oder seiner dynamischen Programmiervariante, dem Inside-Outside-Algorithmus, zu trainieren. Dies geschieht entlang der in Carroll und Rooth (1998) präsentierten Ideen, die eine lexikalisierte kontextfreie Grammatik für Englisch entwickelten und mit einer Version des Inside-Outside-Algorithmus auf einem unannotierten Textkorpus, dem *British National Corpus*, trainierten und evaluierten.

Der Schlüsselschritt, um den Inside-Outside-Algorithmus auf lexikalisierte kontextfreie Grammatiken anwenden zu können, besteht in einer relativ einfachen *Grammatiktransformation*, mit der jede lexikalisierte kontextfreie Grammatik auf eine gewöhnliche kontextfreie Grammatik zurückgeführt werden kann.

Zunächst werden anhand eines Beispiels die Begriffe *lexikalisierte kontexfreie Grammatik* und *lexikalisierter Syntaxbaum* vorgestellt.

Der in Abbildung 5.2 gezeigte Syntaxbaum gehört zu der kontextfreien Grammatik aus Abbildung 5.3. Schon im Abschnitt 2.2 wurde vorgestellt, dass die Wahrscheinlichkeit

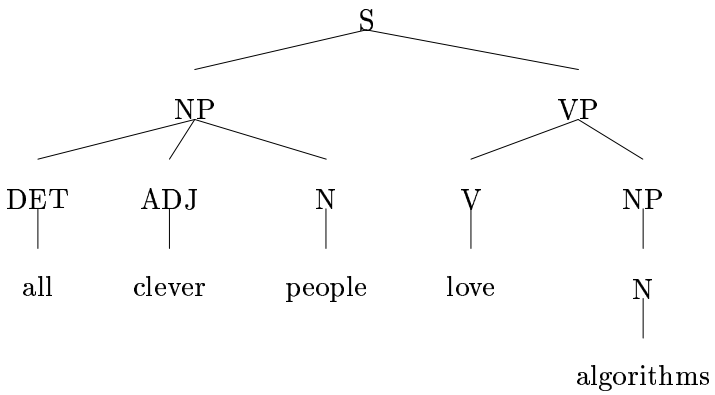


Abbildung 5.2: Kontextfreier Syntaxbaum

S	→	NP VP
NP	→	DET N
NP	→	DET ADJ N
NP	→	N
VP	→	V NP
DET	→	all
DET	→	the
ADJ	→	clever
N	→	people
N	→	algorithms
V	→	love
V	→	kill

Abbildung 5.3: Kontextfreie Grammatik

S	→	NP VP'
NP	→	DET N'
NP	→	DET ADJ N'
NP	→	N'
VP	→	V' NP
DET	→	all
DET	→	the
ADJ	→	clever
N	→	people
N	→	algorithms
V	→	love
V	→	kill

Abbildung 5.4: Lexikalisierte kontextfreie Grammatik

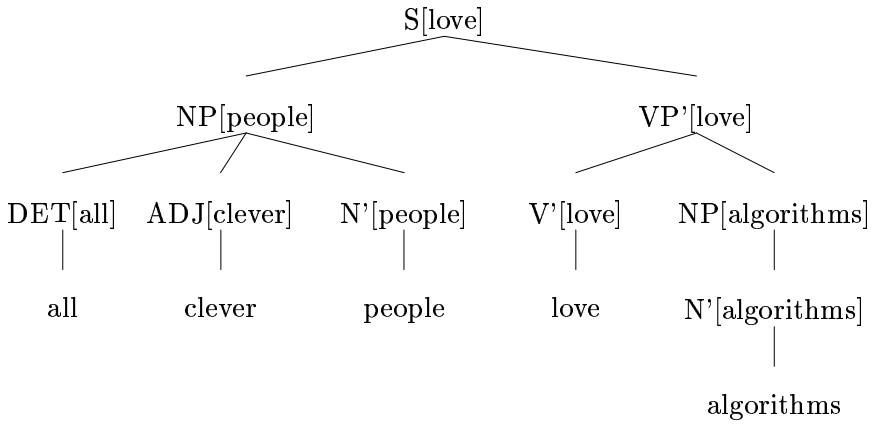


Abbildung 5.5: Lexikalisierter Syntaxbaum

eines kontextfreien Syntaxbaums das Produkt aller seiner Grammatikregel-Wahrscheinlichkeiten ist.

Durch eine Lexikalisierung der Grammatikregeln erhält man eine *lexikalisierte kontextfreie Grammatik*, die in Abbildung 5.4 gezeigt ist. Die *Kopfmarkierungen* (') der Grammatikregeln der lexikalisierten kontextfreien Grammatik dienen dazu, lexikalische Köpfe zu verwalten. Natürlich benötigen Grammatikregeln, deren rechte Seite aus einem Wort des Grammatiklexikons bestehen, keine Kopfmarkierung.

Abbildung 5.5 zeigt einen *lexikalisierten Syntaxbaum*. Die lexikalischen Köpfe, die in den Syntaxbäumen im folgenden mit eckigen Klammern [.] notiert werden, können hierbei *bottom-up* – gemäss der Kopfmarkierungen (') der lexikalisierten kontextfreien Grammatik – verwaltet werden.

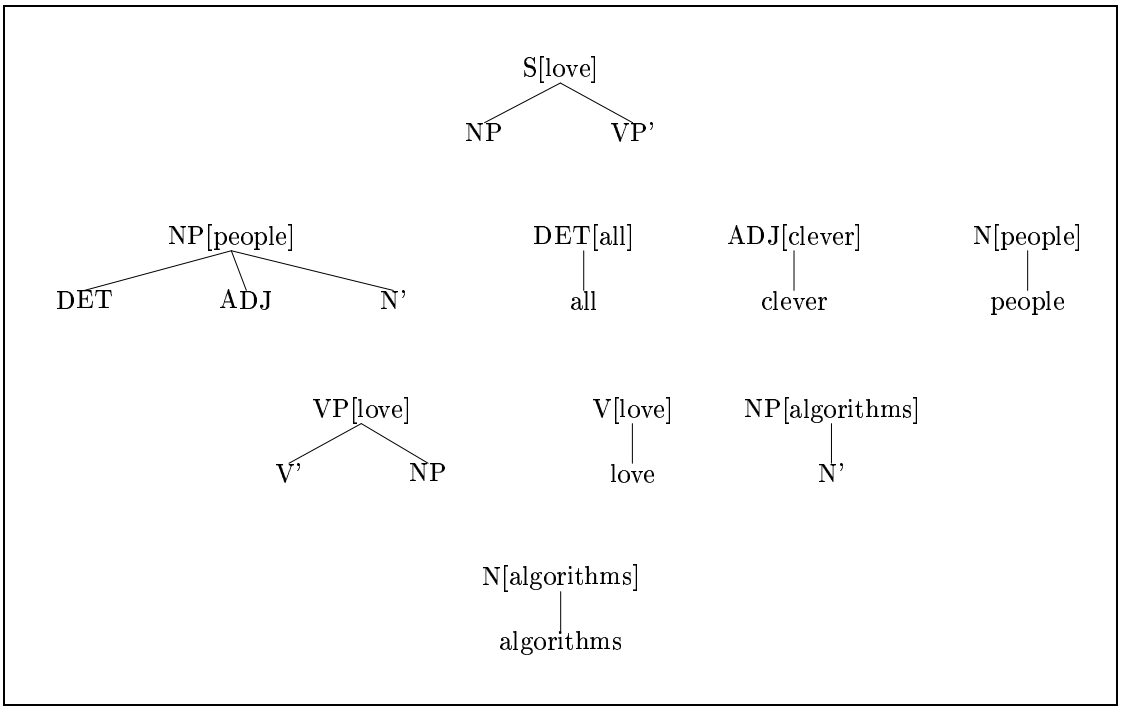


Abbildung 5.6: Lexikalisch annotierte Grammatikregeln

Die *Wahrscheinlichkeit eines lexikalisierten Syntaxbaums* ist das Produkt der Wahrscheinlichkeiten aller im Syntaxbaum vorkommenden *lexikalisch annotierten Grammatikregeln* und der dort getroffenen *lexikalischen Entscheidungen*[2]. Diese beiden neuen Begriffe werden im folgenden geklärt.

Die im lexikalisierten Syntaxbaum vorkommenden Grammatikregeln der lexikalisierten Grammatik, wobei am Mutterknoten – nicht aber an den Töchterknoten – zusätzlich der lexikalische Kopf annotiert wird, werden *lexikalisch annotierte Grammatikregeln* genannt. Der lexikalisierte Syntaxbaum aus Abbildung 5.5 hat die in Abbildung 5.6 gezeigten lexikalisch annotierten Grammatikregeln.

Die *lexikalischen Entscheidungen* sind die in einem lexikalisierten Syntaxbaum vorkommenden grammatischen Relationen, wobei die beteiligten Mutter- und Töchterknoten mit ihren lexikalischen Köpfen annotiert werden. Als *grammatische Relation* soll in diesem Zusammenhang jede im lexikalisierten Syntaxbaum vorkommende Relationen zwischen Mutter- und Tochterknoten bezeichnet werden, wobei einzig die Relation zwischen einem Mutterknoten und dem die Kopfmarkierung (') tragenden Tochterknoten *nicht* als grammatische Relation aufgefasst wird. Abbildung 5.7 zeigt die lexikalischen Entscheidungen des lexikalisierten Syntaxbaums aus Abbildung 5.5.

[2]multipliziert mit der Wahrscheinlichkeit $p(S[v])$ des lexikalischen Satzkopfes (siehe unten).

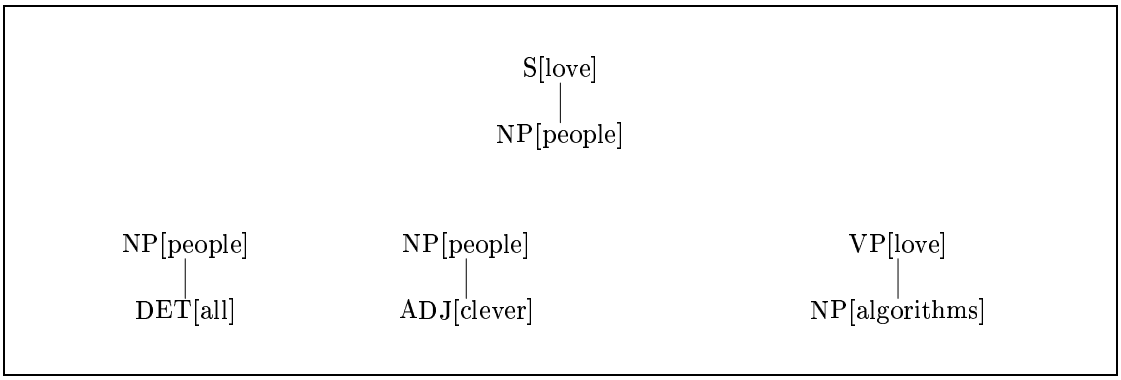


Abbildung 5.7: Lexikalische Entscheidungen

Der EM-Algorithmus für lexikalisierte kontextfreie Grammatiken

Eine lexikalisierte kontextfreie Grammatik kann durch die folgende Grammatiktransformation auf eine (unlexikalisierte) kontextfreie Grammatik zurückgeführt werden (vergleiche Carroll und Rooth (1998)):

Grammatiktransformation zur Zurückführung einer lexikalisierten kontextfreien Grammatik auf eine unlexikalisierte kontextfreie Grammatik: Das Startsymbol S und das Grammatiklexikon $\mathcal{L}ex$ der gegebenen lexikalisierten kontextfreien Grammatik werden unverändert übernommen. Die Grammtikkategorien der transformierten kontextfreien Grammatik sind von der Form $Y(X[v])$ oder $X[v]$. Hierbei sind X, Y zwei beliebige Grammatikkategorien der gegebenen lexikalisierten kontextfreien Grammatik und $v \in \mathcal{L}ex$ ein Wort aus dem Grammatiklexikon. Die Grammatikregeln der transformierten kontextfreien Grammatik sind:

- *Startsymbol*: für jedes Wort $v \in \mathcal{L}ex$ aus dem Grammatiklexikon:

$$S \rightarrow S[v]$$

- *Lexikalisch annotierte Grammatikregeln*: für alle Grammatikregeln

$$X \rightarrow \ \ldots\ X_{i-1}\ \ X_i'\ \ X_{i+1}\ \ldots$$

der gegebenen lexikalisierten kontextfreien Grammatik und für jedes Wort $v \in \mathcal{L}ex$ aus dem Grammatiklexikon:

$$X[v] \rightarrow \ \ldots\ X_{i-1}(X[v]) \quad X_i[v] \quad X_{i+1}(X[v])\ \ldots$$

- *Lexikalische Entscheidungen*: für jede neue Grammatikkategorie $Y(X[v])$ und für jedes Wort $w \in \mathcal{L}ex$ aus dem Grammatiklexikon:

$$Y(X[v]) \rightarrow X[w]$$

- *Lexikoneinträge*: für jede Grammatikregel $X \rightarrow v$ der gegebenen lexikalisierten kontextfreien Grammatik mit einem Wort $v \in \mathcal{L}ex$ aus dem Grammatiklexikon:

$$X[v] \rightarrow v$$

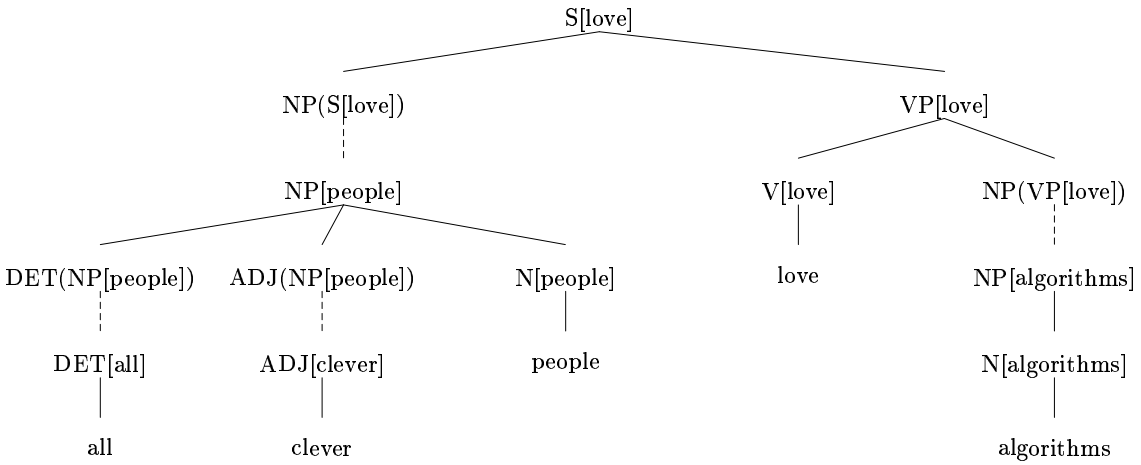
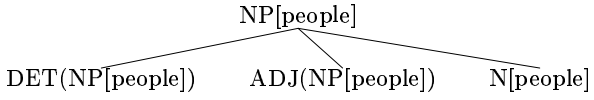


Abbildung 5.8: transformierter Syntaxbaum

Nach der Grammatiktransformation erhält man den den *transformierten Syntaxbaum* aus Abbildung 5.8. Da der transformierte Syntaxbaum die relativ komplexen Grammatikkategorien (etwa DET(NP[people]) oder NP(VP[love])) der transformierten kontextfreien Grammatik benutzt, ist er relativ unübersichtlich. Trotdem ist der transformierte Syntaxbaum ein "normaler" Syntaxbaum einer "normalen" kontextfreien Grammatik. Seine Wahrscheinlichkeit errechnet sich daher als das Produkt der Wahrscheinlichkeiten aller in ihm vorkommenden Grammatikregeln. Der letzte Schritt ist daher, zu zeigen, dass sich die Grammatikregeln des transformierten Syntaxbaums *exakt* auf die lexikalisch annotierten Grammatikregeln und die lexikalischen Entscheidungen des lexikalisierten Syntaxbaums abbilden lassen.

Die Abbildungen 5.9 und 5.10 zeigen sämtliche Grammatikregeln des transformierten Syntaxbaums. Offensichtlich korrespondiert ein Teil der Grammatikregeln des transformierten Syntaxbaums mit den *lexikalisch annotierten Grammatikregeln* des lexikalisierten Syntaxbaums, während der andere Teil mit den *lexikalischen Entscheidungen* des lexikalisierten Syntaxbaums korrespondiert. Offensichtlich ist diese Korrespondenz sehr eng, da es einen Algorithmus gibt, der aus den Grammatikregeln des transformierten Syntaxbaums die lexikalisch annotierten Grammatikregeln (oder die lexikalischen Entscheidungen) des lexikalisierten Syntaxbaums konstruiert und umgekehrt. Wenn man einmal aus der folgenden Grammatikregel



des transformierten Syntaxbaums die lexikalisch annotierte Grammatikregel

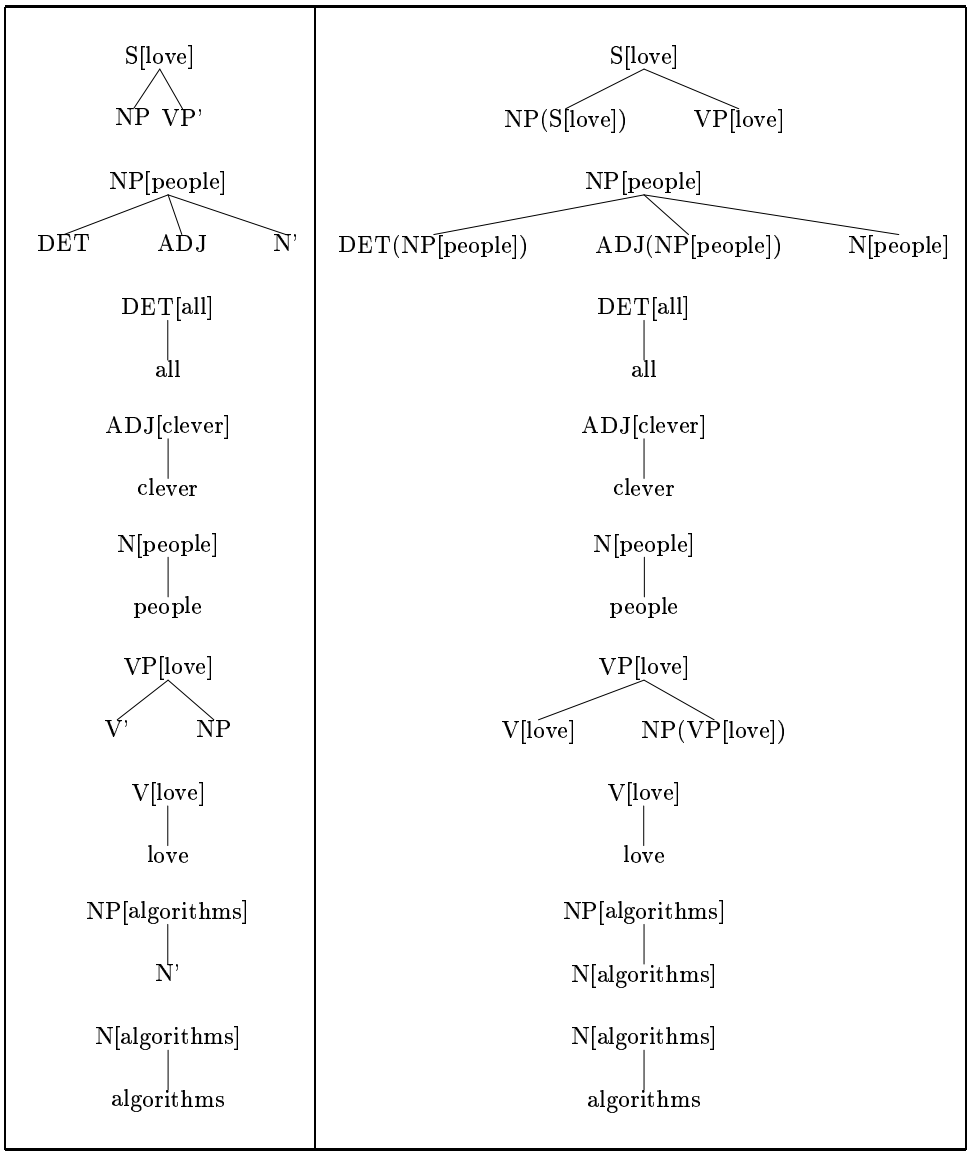


Abbildung 5.9: Lexikalisch annotierte Grammatikregeln des lexikalisierten Syntaxbaums (links) und korrespondierende Grammatikregeln des transformierten Syntaxbaums (rechts)

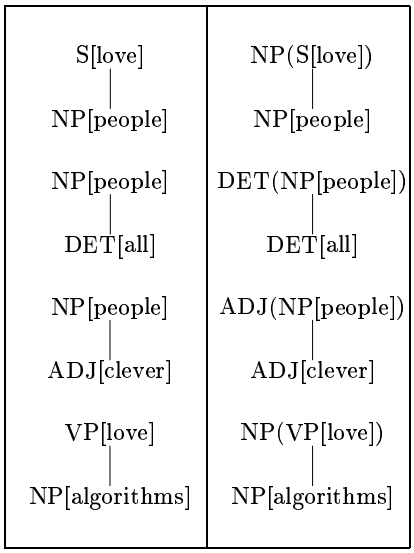
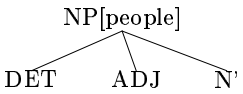


Abbildung 5.10: Lexikalische Entscheidungen des lexikalisierten Syntaxbaums (links) und korrespondierende Grammatikregeln des transformierten Syntaxbaums (rechts)



konstruiert hat[3], so sieht man auch die allgemeine Verfahrensweise ein.

Natürlich können die Grammatikregel-Wahrscheinlichkeiten der transformierten Grammatik – da es sich bei dieser um eine (mehr oder weniger gewöhnliche) kontextfreie Grammatik handelt – mit dem EM-Algorithmus für kontexfreie Grammatiken, bzw. mit seiner dynamischen Programmiervariante, dem Inside-Outside-Algorithmus estimiert werden. Bei dieser Re-Estimierung wird angenommen, dass sich die Wahrscheinlichkeiten der Grammatikregeln mit derselben linken Seite auf Eins aufsummieren. Für die transformierte kontextfreie Grammatik heisst dies:

[3]Der Mutterknoten wird unverändert übernommen. Aus den Töchtern wird zunächst der lexikalische Kopf ("[people]") gelöscht. Die Tochter, die als reine Kategorie übrigbleibt ("N") wird mit der Kopfmarkierung (') versehen. Aus den restlichen Töchtern wird die Information über die Mutterkategorie ("(NP)") gelöscht. Für Grammtikregeln, die mit lexikalischen Entscheidungen korrespondieren, gibt es einen ähnlichen Algorithmus.

Standard-Nebenbedingung für kontexfreie Grammatiken nach Booth und Thompson (1973), angewandt auf die transformierte kontexfreie Grammatik:

- *Startsymbol*:
$$\sum_{v \in \mathcal{L}ex} p(S \rightarrow S[v]) = 1 \ .$$
- *Lexikalisch annotierte Grammatikregeln*: für alle Grammatikkategorien X und alle Wörter $v \in \mathcal{L}ex$ des Grammatiklexikons gilt:
$$\sum_{\alpha} p(X[v] \rightarrow \alpha) = 1 \ .$$
- *Lexikalische Entscheidungen*: für alle Grammatikkategorien X, Y und alle Wörter $v \in \mathcal{L}ex$ des Grammatiklexikons gilt:
$$\sum_{w \in \mathcal{L}ex} p(Y(X[v]) \rightarrow X[w]) = 1 \ .$$

Die Standard-Nebenbedingung der transformierten Grammatik wird daher zu einer Standard-Nebenbedingung für die Wahrscheinlichkeitsverteilungen lexikalisierter kontexfreier Grammatiken.

Standard-Nebenbedingung für die Wahrscheinlichkeitsverteilungen einer lexikalisierten kontextfreien Grammatik:

- Es liegt eine Wahrscheinlichkeitsverteilung über die Satzköpfe vor (siehe Fussnote):
$$\sum_{v \in \mathcal{L}ex} p(S[v]) = 1 \ .$$
- Für alle Grammatikkategorien X und alle Wörter $v \in \mathcal{L}ex$ des Grammatiklexikons gilt:
$$\sum_{\alpha} p(X[v] \rightarrow \alpha) = 1 \ .$$
Hierbei bezeichnet $X[v] \rightarrow \alpha$ die *lexikalisch annotierte Grammatikregel*, die aus der Grammatikregel $X \rightarrow \alpha$ besteht, wobei an die Mutterkategorie X der lexikalische Kopf v annotiert ist.
- Für alle Grammatikkategorien X, Y und alle Wörter $v \in \mathcal{L}ex$ des Grammatiklexikons gilt:
$$\sum_{w \in \mathcal{L}ex} p(\ X[v]Y[w]\ ) = 1 \ .$$
Hierbei bezeichnet $X[v]Y[w]$ eine *lexikalisch annotierte grammatische Relation* zwischen der Mutterkategorie X, annotiert mit dem lexikalischen Kopf v, und der Tochterkategorie Y, annotiert mit dem lexikalischen Kopf w.

Im nächsten Theorem werden die Ergebnisse zusammengefasst:

Theorem (Probabilistische lexikalisierte kontextfreie Grammatiken):

1. Jede *lexikalisierte kontextfreie Grammatik* kann mit der oben angegebenen *Grammatiktransformation* in eine "gewöhnliche" kontextfreie Grammatik übertragen werden, sodass (i) die lexikalisierten Syntaxbäume und die transformierten Syntaxbäume, aber auch (ii) die Wahrscheinlichkeitsparameter der lexikalisierten Syntaxbäume (d.h. die Wahrscheinlichkeiten der lexikalisch annotierten Grammatikregeln, der lexikalischen Entscheidungen, sowie die der Satzköpfe) und die Wahrscheinlichkeitsparameter der transformierten Syntaxbäume (ihre "gewöhnlichen" Grammatikregel-Wahrscheinlichkeiten) eineindeutig aufeinander abbildbar sind.

2. Lexikalisierte kontextfreie Grammatiken, die die oben genannten Standard-Nebenbedingungen erfüllen, können mit dem EM-Algorithmus oder seiner dynamischen Programmiervariante, dem Inside-Outside-Algorithmus, trainiert werden. Lexikalisierte kontextfreie Grammatiken weisen hierbei dieselben stochastischen Eigenschaften auf, wie "normale" kontextfreie Grammatiken, z.B. erhöht sich ihre Log-Likelihood während des Trainings mononton und wird möglicherweise gegen ein lokales Maximum der Log-Likelihood konvergieren.

Beweis: (2.) folgt aus (1.) und den Ergebnissen aus Kapitel 3. (1.) wurde bereits informell gezeigt. **q.e.d.**

Mit diesem schönem Ergebnis soll dieser Abschnitt beendet werden.

5.2 Experimente mit deutschen Verbletztsätzen

In diesem Abschnitt wird gemeinsame Arbeit mit Franz Beil, Glenn Carroll, Stefan Riezler und Mats Rooth vorgestellt. Das in Abschnitt 5.1 vorgestellte Wahrscheinlichkeitsmodell für lexikalisierte kontextfreie Grammatiken wurde in Charniak (1995) und Carroll und Rooth (1998) eingeführt. Im folgenden wird ein Experiment beschrieben, in dem eine leicht veränderte Version des Modells von Carroll und Rooth für Deutsch angewendet wird. Deutsch ist eine Sprache mit reicher Flektionsmorphologie und freier Wortfolge, bzw. im Vergleich zum Englischen mit freierer Phrasenfolge. Es werden daher Techniken verwendet, die es erlauben, den Inside-Outside-Algorithmus auf eine lexikalisierte kontextfreie Grammatik einer solchen Sprache anzuwenden. Diese Techniken werden es ermöglichen, sowohl Argumente zu tilgen, als auch mit veränderter Satzordnung umzugehen (Scrambling), Kasus mit Kategorie-Features auszudrücken, Kategorienamen wie Nomen, Artikel, Adjektive, aber auch ihre Projektionen zu verwenden, sowie die Lexikalisierung auf der Basis von nichtflektierten Lemmas anstelle von Wortformen vorzunehmen.

Korpus und Morphologie

Die Daten des Experiments bestehen aus einem deutschen Korpus von Nebensätzen, die aus einem 200 Million Wort grossen Zeitungskorpus mit Hilfe einer regulären Grammatik herausgefiltert wurden. Die Sätze sind zwischen 4 und 12 Worten lang. Neben den verbalen Argumenten aus infiniten VPs, gibt es keine weiteren Satzeinbettungen. Alle Sätze enthalten keine Interpunktion, ausser dem Punkt am Satzende. Der Korpus besteht aus 4 128 873 Worten, bzw. 450 526 Sätzen, was 9.2 Worte pro Satz ergibt. Die Worte wurden automatisch mit einer Liste von Part-of-Speech (PoS) Tags versehen. Dazu wurde ein Morphologie-System verwendet, das auf endlichen Automaten Karttunen et al. (1994), Schiller und Stöckert (1995) basiert.

Bei einer flektionsreichen Sprache wie dem Deutschen besteht ein praktisches Problem für den Inside-Outside Algorithmus in der grossen Anzahl von zu bewältigenden lexikalischen Köpfen und präterminalen Grammatikkategorien, die aus morpho-syntaktischen Wortmerkmalen resultieren. Neben den Hauptklassen, wie Nomen, Adjektive usw., weist der Tagger jedem ambigen Wort eine Liste von möglichen Kombinationen der Flektionsmerkmale Geschlecht, Person und Numerus zu. Der obere Teil der Abbildung 5.11 zeigt als Beispiel ein nominales bzw. adjektivisches PoS Tag; dies wird durch das PoS Tag mit dem Zeichen '+' ausgedrückt. Um die Parameteranzahl zu reduzieren, und um die Anzahl der Parsewälder zu minimieren, die beim Inside-Outside Algorithmus verwendet werden, wurden die flektierten Lesarten von Adjektiven, adjektivische Nomen, Artikeln und Pro-

```
analyze> Deutsche
1. deutsch^ADJ.Pos+NN.Fem.Akk.Sg
2. deutsch^ADJ.Pos+NN.Fem.Nom.Sg
3. deutsch^ADJ.Pos+NN.Masc.Nom.Sg.Sw
4. deutsch^ADJ.Pos+NN.Neut.Akk.Sg.Sw
5. deutsch^ADJ.Pos+NN.Neut.Nom.Sg.Sw
6. deutsch^ADJ.Pos+NN.NoGend.Akk.Pl.St
7. deutsch^ADJ.Pos+NN.NoGend.Nom.Pl.St
8. *deutsch+ADJ.Pos.Fem.Akk.Sg
9. *deutsch+ADJ.Pos.Fem.Nom.Sg
10. *deutsch+ADJ.Pos.Masc.Nom.Sg.Sw
11. *deutsch+ADJ.Pos.Neut.Akk.Sg.Sw
12. *deutsch+ADJ.Pos.Neut.Nom.Sg.Sw
13. *deutsch+ADJ.Pos.NoGend.Akk.Pl.St
14. *deutsch+ADJ.Pos.NoGend.Nom.Pl.St

==>   Deutsche  { ADJ.E, NNADJ.E }
```

Abbildung 5.11: Zusammenfassung morphologischer Features

nomen in einem einzigen morphologischen Feature zusammengefasst (siehe das Beispiel in Abbildung 5.11). Dies reduzierte die Anzahl der Präterminal-Kategorien. Die Beispiele in Abbildung 5.12 verdeutlichen die Zusammenfassung der Lesarten: beispielsweise kann das Wort *das* sowohl ein Artikel als auch ein Demonstrativpronomen sein; *westdeutschen* hat eine adjektivische Lesart mit dem morphologischen Feature `N`, das durch das Flektionssuffix angezeigt wird. Das spezielle Tag `UNTAGGED` wird verwendet, um zu zeigen, dass der Tagger diesem Wort kein Tag zuweist. Ein Grossteil der `UNTAGGED` Worte sind Eigennamen, die nicht als solche erkannt wurden. Diese Lücke in der Morphologie hat jedoch fast keine Auswirkung auf das Experiment.

während	{ ADJ.Adv, ADJ.Pred, KOUS, APPR.Dat, APPR.Gen }
sich	{ PRF.Z }
das	{ DEMS.Z, ART.Def.Z }
Preisniveau	{ NN.Neut.NotGen.Sg }
dem	{ DEMS.M, ART.Def.M }
westdeutschen	{ ADJ.N }
annähere	{ VVFIN }
.	{ PER }

Abbildung 5.12: Corpus Clip

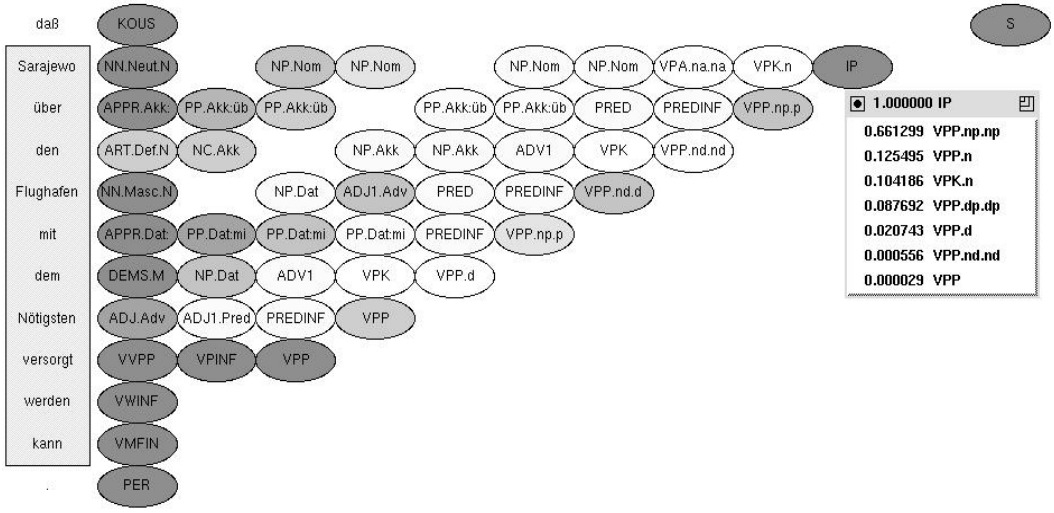


Abbildung 5.13: Chart-Browser

Satz: *'dass Sarajevo über den Flughafen mit dem Nötigsten versorgt werden kann'*

Grammatik

Die Grammatik ist eine manuell entwickelte lexikalisierte kontext-freie Phrasenstruktur-Grammatik für deutsche Nebensätze mit 5508 Regeln und 562 Kategorien, von denen 209 terminale Kategorien sind. Der Formalismus wurde von Carroll und Rooth (1998) entwickelt. Eine lexikalisierte Grammatikregel ist von der Form:

$$\text{Mutterkategorie} \rightarrow \text{Nicht-Köpfe Kopf' Nicht-Köpfe (Freq)}$$

Die Köpfe der Regeln sind mit einem Apostroph gekennzeichnet. Die Folge der Nicht-Köpfe können leer sein. `Freq` ist die erwartete Häufigkeit der Regel, die randomisiert initialisiert wird und durch den Inside-Outside Algorithmus re-estimiert wird. Zur Behandlung von systematischen Mustern, die sich auf Features beziehen, wurden die Regeln automatisch mit Lisp generiert und nicht von Hand geschrieben. Es gibt nur wenige Ausnahmen (z.B. Regeln für die Koordination, die S-Regel), in denen die Regeln mehr als zwei Töchter haben. Die Grammatikentwicklung wurde durch einen Chart-Browser unterstützt, der es erlaubt, schnell und effizient Grammtikfehler (Carroll 1997a) zu entdecken.

Abbildung 5.13 zeigt, dass die Ambiguität in der Chart ziemlich hoch ist, obwohl die Grammatik und der Korpus begrenzt sind. Ein Satz des Korpus hat durchschnittlich 9202 Baumanalysen. In den Zellen des Chart-Browsers ist die wahrscheinlichste Kategorie der Teilanayse und ihre estimierte Häufigkeit zu finden. Das Fenster unterhalb von `IP` stellt die nach Wahrscheinlichkeit geordnete Liste aller möglichen Kategorien für eine Teilanalyse dar. Die Regeln (Chart-Ränder) mit ihren Häufigkeiten können mit einem

zusätzlichen Menü gezeigt werden. In dem Chart-Browser werden Farben benutzt, um die Häufigkeit darzustellen, die mit dem Inside-Outside Algorithmus berechnet wurden, sodass Grammatik- und Estimierungsfehler ohne grossen Aufwand erkannt werden können. Die Satzabdeckung der Grammatik liegt bei 88,5% und die Token-Abdeckung, die im Korpus vorkommen, bei 87,9%. Parsingfehler treten in 6,6% der Fälle bei `UNTAGGED` Worten auf, ca. 1,3% sind infinitivische Konstruktionen und ca. 1,6% sind Koordinationen, die nicht von der Grammatik analysiert werden können.

Kasusfeature und Agreement

Zusätzlich zu den vier nominalen Kategorien Nominativ, Genetiv, Dativ und Akkusativ werden Features mit disjunkter Interpretation verwendet, wie beispielsweise `Dir` für Nominativ und Dativ. Die Grammatik ist so geschrieben, dass nicht-disjunkte Feature möglichst früh eingeführt werden, was sich positiv auf die Grösse des Parsewaldes auswirkt. Durch die Zusammenfassung mancher Parameter (Pooling) verringert sich deren Gesamtanzahl, was die Parameter-Estimierung erleichtert. Ausserdem erzwingt diese Technik in gewissen Fällen ein Agreement über den ganzen Baum hinweg. Agreement zwischen nominativischen NPs und dem gebeugten Verb (z.B. Numerus) wird dagegen durch die Grammatik nicht gefordert, um die Anzahl der Parameter und Regeln zu kontrollieren. Für nominale Phrasen wird die Idee von Abneys Chunk-Grammatik (Abney 1996) angewendet. Das Nomen-Chunk (NC) ist eine Annäherung an die nicht-rekursive Projektion, die zum einen Argumente ausschliessen, die nach dem Kopf stehen, aber auch adverbiale Adjunkte, die schon vor den Modifikatoren und Artikeln des Kopfes eingeführt wurden. Allerdings beinhalten sie partizipiale Modifikatoren mit ihren Komplementen, die vor dem Kopf stehen können. Da kontext-freies Parsen, Parsewälderkonstruktion und Inside-Outside Algorithmus verwendet werden, werden Chunks (im Gegensatz zu Abneys Arbeit) nicht durch deterministisches Parsen erzeugt. Vielmehr wird bei der Grammatik-Entwicklung ein graphisches Debugging dazu verwendet, um möglichst lange Konstituenten mit möglichst hoher estimierter Häufigkeit zu generieren.

Subkategorisierungsrahmen von Verben

Die Grammatik unterscheidet zwischen vier Klassen von Subkategorisierungsrahmen: aktiv (VPA), passiv (VPP), infinitivische Rahmen (VPI) und Kopulakonstruktionen (VPK). Die möglichen Argumente der Verbrahmen sind nominativische (n), dativische (d) und akkusativische (a) NPs, Reflexivpronomen (r), Präpositionalphrasen, d.h. PPs (p), und infinitivische VPs (i). Die Grammatik unterscheidet nicht zwischen einfachen infinitvischen

class	#	frame types
VPA	15	n, na, nad, nai, nap, nar, nd, ndi, ndp, ndr, ni, nir, np, npr, nr
VPP	13	d, di, dp, dr, i, ir, n, nd, ni, np, p, pr, r
VPI	10	a, ad, ap, ar, d, dp, dr, p, pr, r
VPK	2	i, n

Abbildung 5.14: Anzahl und Typen deutscher Verbrahmen

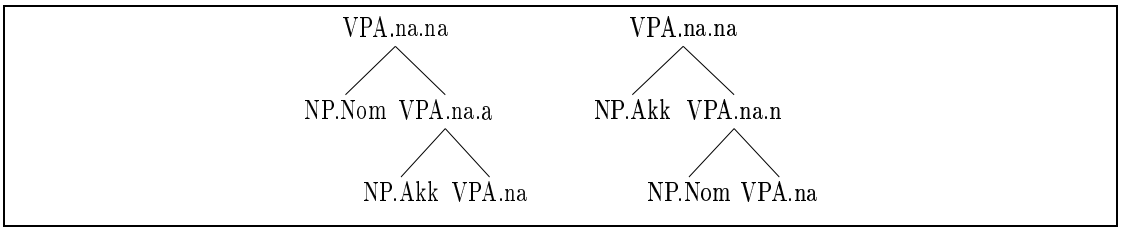


Abbildung 5.15: Kodierung einer kanonischen und einer gescrambelten Argumentordnung

VPs und *zu*-infinitivischen VPs. Die Grammatik ist so konstruiert, dass sie zwischen unterschiedlichen PPs unterscheiden kann (in Abhängigkeit des präpositionalen Kopfs der PP). Die Anzahl und die Verbrahmentypen der verschiedenen Rahmenklassen sind in Abbildung 5.14 dargestellt. Deutsch ist eine Sprache mit einer verhältnismässig freien Wortfolge, die das Verschieben von Argumenten erlaubt (Scrambling). Scrambling zeigt sich in der besonderen Reihenfolge der Argumente, die den Verbrahmen sättigen. Abbildung 5.15 zeigt ein Beispiel einer kanonischen Anordnung von Subjekt-Objekt in einem aktivischen transitiven Rahmen und eine gescrambelte Ordnung, in der das Objekt vor dem Subjekt steht. Die Anzahl der Grammatikregeln erhöht sich durch die Möglichkeit, dass Verbargumente gescrambelt werden können (z.B. gibt es 102 Möglichkeiten, die Argumentregeln in allen VPA-Rahmen zu kombinieren). Adverbien und nicht-subkategorisierte PPs werden als Adjunkte einer VP-Kategorie analysiert, die dadurch keine Subkatrahmen-Position sättigt. In früheren Experimente wurde eine flache Satzstruktur, mit Regeln für die Vertauschung von Komplementen verwendet. Da die Anzahl der Rahmen zunahm, führte dies zu vielen und auch unerlaubten Regeln, speziell zu Regeln, die Adjunkte einschlossen.

Parameter

Die Parametrisierung entspricht, mit einer einzigen Modifikation, der in Abschnitt 5.1 vorgestellten Parametrisierung aus den Wahrscheinlichkeiten für *lexikalisch annotierte Grammatikregeln* und *lexikalischen Entscheidungen*. Die Änderung ist, dass die Lexikalisierung

durch nicht-flektierte Lemmata erzeugt wird, anstatt durch die Wortform selbst. Diese Vorgehensweise reduziert die Anzahl der lexikalischen Parameter und resultiert in akzeptableren Modelgrössen. Ausserdem wird die Aufteilung estimierter Häufigkeiten auf die flektierten Formen verhindert. Flektierte Formen werden an den Blättern der Bäume erzeugt, bedingt durch die Terminalkategorie und das Lemma. Faktisch ergibt sich hieraus eine dritte Parameterfamilie, obwohl im Normalfall die Auswahl der flektierten Formen deterministisch ist.

Parameterpooling wird realisiert, indem alle Elternkategorien der Form VP.x.y auf die Kategorie VP.x abgebildet werden, um so die Wahrscheinlichkeiten der lexikalischen Entscheidungen zu definieren. Es folgt, dass z.B. eine akussativische Tochter eines Nominativ-Akkusativ-Verbs die gleichen Wahrscheinlichkeiten für lexikalische Entscheidungen verwendet, unabhängig davon, ob es sich um eine Default- oder eine gescrambelte Wortordnung handelt. Im Gegensatz zu den lexikalischen Entscheidungen, werden die Wahrscheinlichkeiten der lexikalisch annotierten Grammatikregel nicht gepoolt. Die estimierten Regel-Häufigkeiten werden daher aufgeteilt, wenn die Elternkategorie ein Flektionsfeature trägt.

Estimierung

Das Training einer probabilistischen CFG erfolgt in drei Schritten: (i) nicht lexikalisiertes Training mit dem `supar`-Parser, (ii) Bootstrapping eines lexikalisierten Modells aus den trainierten nicht-lexikalisierten Modellen mit dem `ultra`-Parser, und zuletzt (iii) lexikalisiertes Training mit dem `hypar`-Parser (Carroll 1997b). Jeder der drei Parser verwendet den Inside-Outside-Algorithmus. `supar` verwendet nicht-lexikalisierte Bäume und deren Wahrscheinlichkeiten, während `ultra` und `hypar` die lexikalisierten Bäume und deren Wahrscheinlichkeitsmodell (siehe Abschnitt 5.1) unterstützen. `ultra` und `hypar` sammeln die Frequenzen der lexikalisierten Regeln und auch die lexikalisierten Entscheidungen auf, während `supar` nur die nicht-lexikalisierten Regelhäufigkeiten sammelt.

Die Experimente zeigen, dass es sich lohnt, am Anfang mit einem nicht-lexikalisierten Modell zu trainieren. Trotz einer manuell entwickelten Grammatik, die kein Pruning der überflüssigen Regeln benötigt, wie eine automatisch generierte Grammatik, ist das lexikalisierte Modell beträchtlich besser, wenn vorher nicht-lexikalisiert trainiert wurde (siehe auch Ersan und Charniak (1995), die ähnliche Beobachtungen machten). Der Vergleich zwischen einem ausschliesslich lexikalisierten Training (ohne vorheriges Trainieren eines nicht-lexikalisierten Modells) und dem hier verwendeten Standard-Trainingsverfahren, das ein vorheriges nicht-lexikalisiertes Training beinhaltet, spricht für dieses Trainingsregime (vergleiche die verschiedenen 'lex0' und 'lex2' Kurven in Abbildung 5.18 und 5.19). Jedoch

A		B		C	
1:	52.0199	1:	53.7654	1:	49.8165
2:	25.3652	2:	26.3184	2:	23.1008
3:	24.5905	3:	25.5035	3:	22.4479
⋮	⋮	⋮	⋮	⋮	⋮
13:	24.2872	55:	25.0548	70:	22.1445
14:	24.2863	56:	25.0549	80:	22.1443
15:	24.2861	57:	25.0549	90:	22.1443
16:	24.2861	58:	25.0549	95:	22.1443
17:	24.2867	59:	25.055	96:	22.1444

Abbildung 5.16: Übertraing (Iterative Messung der Cross-Entropie auf Heldout-Daten)

muss die Anzahl der nicht-lexikalisierten Trainingsiterationen kontrolliert werden.

Ein Standard-Verfahen, um *Übertraining* zu messen, ist die Messung der Log-Likelihood Werte auf sogenannten *Held-Out-Daten*, d.h. auf einem Korpus von Sätzen, der extra für diese Messung reserviert wurde. Während der Log-Likelihood-Wert auf den Trainingsdaten theoretisch nach jeder Trainingsiteration konvergiert, weist ein abnehmender Log-Likelihood-Wert auf den Heldout-Daten auf Übertraining hin. Anstelle der Log-Likelihood, wird in dieser Arbeit allerdings die positrive Cross-Entropie gemessen. Abbildung 5.16 zeigt den Vergleich von verschieden Trainings- und Heldout-Daten-Grössen (Training/Heldout): (A) 50k/50k, (B) 500k/500k, (C) 4.1M/500k. Der Übertrainingseffekt zeigt sich in der Tabelle durch das Ansteigen der Cross-Entropie zwischen den letzten beiden Iterationen. Ergebnisse für ein Übertraining bei lexikalisierten Modellen wurden bis jetzt noch nicht beobachtet.

Jedoch zeigt ein Vergleich der Precision- und Recall-Werte auf den verschieden Evaluierungskorpora während des nicht-lexikalisierten Trainings, dass das hier vorgestellte Standard-Kriterium für Übertraining aus linguistischer Sicht zu schlechten Ergebnissen führt. Während ein mehr oder weniger stabiler Precision/Recall- Wert bei linguistisch weniger komplexen Strukturen wie Nomen-Chunks beobachtet werden konnte, führte ein iteratives nicht-lexikalisiertes Training bis zum Übertrainingsgrenzwert zu einem disaströsen Ergebnis für die Evaluierung von komplexeren linguistischen Kategorien. Die Erkennung von Subkat-Rahmen während 60 Iterationen bei nicht-lexikalisiertem Training zeigte einen massiven Abfall der Precision/Recall-Werte ausgehend von der besten zur letzten Iteration. Der Wert fiel sogar unter das Ergebnis der zufällig initialisierten Grammatik (siehe Abbildung 5.19).

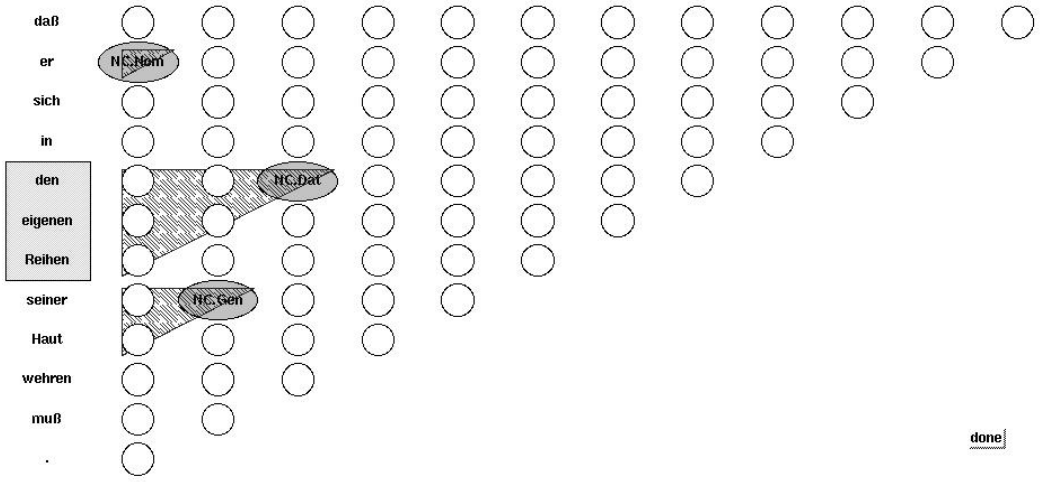


Abbildung 5.17: Chart-Browser für die manuelle Annotation von NCs

Trainingsregime

Das lexikalisierte Training wurde für verschiedene Startpunkte verglichen: ein randomisiertes nicht-lexikalisiertes Modell, das trainierte nicht-lexikalisierte Modell mit den besten Precision/Recall-Ergebnissen, und ein nicht-lexikalisiertes Modell, das nahe an den Cross-Entropie Grenzwert für das Übertraing herankommt. Die Einzelheiten der Trainingsschritte sehen wie folgt aus:

(1) 0, 2 und 60 nicht-lexikalisierte Parsing-Iterationen mit `supar`;

(2) Lexikalisierung mit `ultra`, indem der gesamte Korpus verwendet wird;

(3) 23 lexikalisierte Parsing-Iterationen mit `hypar`.

Das Training wurde auf vier Computern durchgeführt (zwei 167 MHz UltraSPARC und zwei 296 MHz SUNW UltraSPARC-II). Eine Iteration mit `super` auf dem gesamten Korpus mit der beschriebenen Grammatik dauert ungefähr 2.5 Stunden, die Lexikalisierung und die Generierung von lexikalisierten Modellen braucht mehr als 6 Stunden, eine nicht-lexikalisierte Parsing-Iteration ist innerhalb von 5.5 Stunden erledigt.

Evaluierung

Für die Evaluierung wurden insgesamt 600 zufällig ausgewählte Sätze verwendet, die manuell von zwei Personen annotiert wurden. Sie benutzten einen Chart-Browser, um die entsprechenden Zellen mit den Kategorienamen von NCs und maximalen VP-Projektionen zu füllen (vergleiche Abbildung 5.17). Das nachfolgende Alignment der Entscheidungen der Annotierer ergab insgesamt 1353 annotierte NC-Kategorien (mit vier verschiedenen

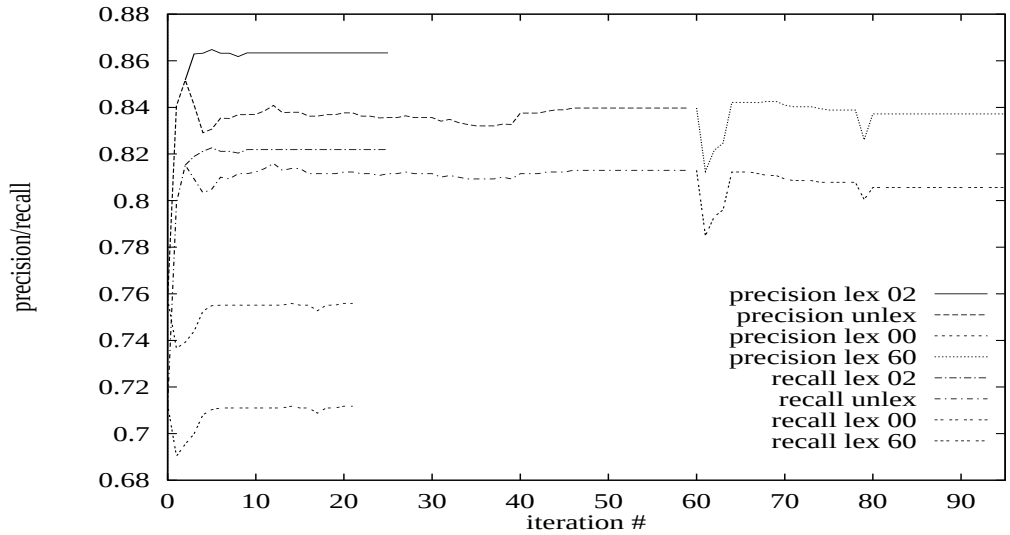


Abbildung 5.18: Evaluierung: Erkennung von Nomen-Chunks mit Kasus

Kasus-Fällen). Die insgesamt 584 gelabelten VP-Kategorien unterteilen sich in 21 verschiedene Verb-Rahmen mit 340 verschiedenen Lemma-Köpfen. Der am häufigsten vorkommende Rahmen ist der aktive transitive Rahmen (164-mal), gefolgt von dem aktiven nicht-transitiven Rahmen (117-mal). Zusammen belegen diese Rahmen 50% aller annotierter Rahmen. 13 Rahmen erscheinen weniger als zehnmal, und fünf nur ein einziges Mal.

Methodik

Um das iterative Training zu evaluieren, wurde nach jeder Iteration der wahrscheinlichste Parse (Viterbi-Parse) für alle 600 Sätze des Testsets extrahiert. Um den wahrscheinlichsten Parse beim nicht-lexikaliserten Training zu extrahieren, wurde Schmids `lopar`-Parser (Schmid 1999b) verwendet. *Precision* und *Recall* wurde gemessen, indem der wahrscheinlichste Parse mit den manuell annotierten Kategorienamen und Spans verglichen wurde. Precision ist die Anzahl der korrekten Vorhersagen, geteilt durch die Anzahl der vorgenommenen Vorhersagen. Recall ist die Anzahl der korrekten Vorhersagen, geteilt durch die Anzahl der von den Annotierern annotierten Phrasen. Die Precision/Recall Ergebnisse, die hier reportiert werden, basieren auf Paaren (Kategoriename, Span) und nicht auf den weniger interessanten Ergebnissen von Spans.

NC Evaluierung

Abbildung 5.18 zeigt die Precision/Recall-Werte für die oben beschriebene Trainingsprozedur, die nach 0, 2 oder 60 unlexikalisierten Trainingsiterationen mit dem lexikalisiertem Training beginnt. Die besten Resultate werden erzielt, indem bereits nach zwei Iterationen unlexikalisiertem Training mit dem lexikalisierten Training begonnen wird. Von 1 353 annotierten NC's mit Kasus wurden 1 103 bzw. 1 112 in dem besten unlexikalisiertem bzw. lexikalisiertem Modell erkannt. Bei einer Anzahl von 1 295 bzw. 1 288 vorgenommenen Rateversuchen des unlexikalisierten bzw. lexikalisierten Modells, bedeutet dies einen Gewinn von 1.2% Precision (85.1% vs. 86.3%) und 0.6% Recall (81.5% vs. 82.1%) durch lexikalisiertes Training. Wie in Abbildung 5.18 gezeigt wird, wird dieser Gewinn bereits in der ersten Iteration ezielt.

Der vergleichsweise kleine Gewinn, der durch lexikalisiertes Training erzielt wurde, kann möglicherweise darauf zurückzuführen sein, dass die Chunking-Aufgabe zu simple ist, um von der lexikalischen Information profitieren zu können.

Die Kurve, die mit "00" bezeichnet ist, gibt die Resultate für lexikalisiertes Training Training wieder, welches von einer zufällig initialisierten Grammatik startet. Die Precision-Werte fallen unter die des unlexikalisierten Zufallsmodels (74%),um nur durch lexikalisiertes Training die Werte des Zufallsmodels zu erreichen (75.6%). Dies könnte darauf hinweisen, dass eine gewisse unlexikalisierte Initialisierung nötig ist, um ein gutes lexikalisiertes Modell trainieren zu können.

(Skut und Brants 1998) reportieren 84.4% Recall und 84.2% für NP- und PP-Chunking ohne Kasus-Annotierung. Obwohl ihre Werte geringfügig schlechter sind, als die hier reportierten, wurden sie auf unbeschränkten Sätzen gemessen.

Evaluierung von Subkategorisierungs-Rahmen

Abbildung 5.19 zeigt die Ergebnisse für die Erkennung von Subkategorisierungsrahmen, die unter denselben Trainingsbedingungen gewonnen wurden. Auch hier werden die besten Resultate nach zwei Iterationen lexikalisierten Trainings erzielt. Von 584 annotierten Verbrahmen im Evaluierungskorpus wurden 384 bzw. 397 mit dem besten unlexikalisiertem bzw. lexikalisiertem Training erkannt. Das beste unlexikalisierte Modell erzielt einen Precision-Wert von 68.4%, der durch das beste lexikalisierte Modell um 2% auf 70.4% angehoben wird. Die Recall-Werte liegen bei 65.7%/68%. Erneut wird der grösste Hinzugewinn in den ersten zwei Iterationen beobachtet.

Die Resultate für lexikalisiertes Training ohne vorangehendes unlexikalisiertes Training

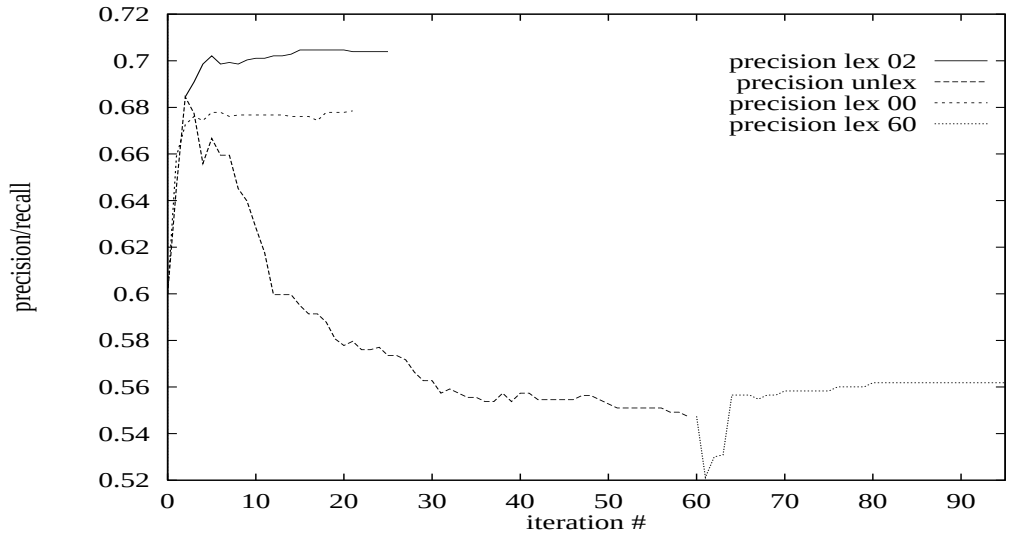


Abbildung 5.19: Evaluierung: Erkennung von Subkategorisierungsrahmen

sind besser als bei der NC-Evaluierung, erreichen allerdings um ca. 2% nicht die besten Resultate.

Die interessanteste Beobachtung ist, dass die Precision-Werte für die Subkatrahmen-Erkennung während des unlexikalisierten Trainings von der zweiten Iteration an stetig fallen. Nach einigen dutzend Iterationen sind die Resultate um 5% schlechter als die des Zufall-Modells und 14% schlechter als die des besten Modells. Die Hauptquelle für diesen Niedergang ist die fehlerhafte Klassifikation von Adjunkt-PP's als Argument-PP's. Z.B. wird die tatsächliche Anzahl von 164 transitiver Rahmen um 76 verfehlt, während der Parser 64 `VPA.nap`-Rahmen vorhersagt, aber nur 12 annotiert wurden. Im Gegensatz hierzu sind die Ergebnisse des lexikalisierten Trainings zumindest nach einigen wenigen Trainingsiterationen stabil.

Die Kurve "lex 60" zeigt die Precision-Werte für lexikalisiertes Training, dass nach 60 Iterationen unlexikalisiertem Training startet. Soweit experimentell bekannt ist, konnte sich das lexikalisierte Training nie von diesem schlechten Ausgangszustand erholen.

Zusammenfassung

Das Hauptresultat dieser Arbeit ist, dass freie Satzordnung, Kasus-Morphologie, Subkategorisierung, sowie NP-internes Genus-, Numerus- und Kasus-Agreement mit einer probabilistischen kontextfreien Grammatik realisiert werden können, indem schon beim Design der Grammatikkategorien und Grammatikregeln darauf geachtet wird, dass Ambiguitäten im Parsewald möglichst spät (in *bottom-up*-Ordnung) aufgelöst werden, um die Grösse

der Parsewälder möglichst klein zu halten. Ausserdem wird eine *Pooling-Technik* benutzt, die die Anzahl der Wahrscheinlichkeitsparameter verringert. Es soll darauf hingewiesen werden, dass in der Grammatik teilweise grössere Kompromisse eingegangen wurden (etwa hinsichtlich des Agreements zwischen Verben und ihren Subjekten), um die Anzahl der Grammatikkategorien, der Grammatikregeln und der Wahrscheinlichkeitsparameter zu kontrollieren.

Ein zweites Resultat ist, dass die lexikalisierten Grammatiken gegenüber den unlexikalisierten Grammatiken nur gering verbesserte Evaluierungsergebnisse aufweisen. Hierfür wurden nicht die beim Baumbank-Trainingsverfahren üblichen "Exact-Match"-Evaluierungen vorgenommen, sondern sorgfältige linguistische Evaluierungen, um satzweise zu messen, wie gut Nominalphrasen und verschiedene Typen von Subkategoriseirungsrahmen von den Grammatiken erkannt werden können.

Ein drittes Ergebnis ist, dass zu viele (mehr als zwei) Iterationen mit dem Inside-Outside-Algorithmus die Ergebnisse der linguistischen Evaluierung verschlechtern. Das gewöhnliche Kriterium, welches im allgemeinen benutzt wird, um Übertraining zu erkennen, nämlich die Messung der Korpuswahrscheinlichkeit eines *Held-Out-Korpus*, wies im Gegensatz zu diesen Ergebnissen an, sehr viel mehr Trainings-Iterationen vorzunehmen, sodass dieses Standard-Kriterium als unangemessen gelten kann.

5.3 Stochastische unifikationsbasierte Grammatiken

In diesem Kapitel wird gemeinsame Arbeit mit Stefan Riezler, Jonas Kuhn und Mark Johnson präsentiert.

Gegenwärtig werden stochastische Grammatiken, die von prinzipiell grösserer Mächtigkeit als die probabilistischen kontextfreien Grammatiken sind, intensiv erforscht. Eine gemeinsame Eigenschaft der meisten dieser Modelle ist die Berücksichtigung von lexikalischen Köpfen in regelbasierten Wahrscheinlichkeitsmodellen, etwa in Collins (1997), Charniak (1997) und Ratnaparkhi (1997). Immer noch offen ist jedoch die Frage, welche Art von Lexikalisierung, z.B. Statistiken für individuelle Wörter oder Statistiken für Wortklassen, benutzt werden sollten.

Ein weiterer gemeinsamer Aspekt dieser Ansätze ist, dass die Wahrscheinlichkeitsmodelle mit Baumbanken trainiert werden, d.h. mit Korpora von manuell annotierten Sätzen. In allen zitierten Ansätzen wird die Penn-Wall-Street-Journal-Baumbank (Marcus et al. 1993) benutzt, was ein Zeichen dafür ist, dass das Baumbank-Training aufwendig ist, da es die erhebliche manuelle Intervention erfordert, Korpora aus speziellen Domänen und speziellen Sprachen mit speziellen Syntaxanalysen zu versehen.

Es wird allgemein angenommen, dass mit dem Training mit dem EM-Algorithmus (Dempster et al. 1977) nur schlechte Resultate erzielbar sind, wenn nicht zumindest partielle Annotationen verwendet werden. Experimentelle Resultate, die dieses "Allgemeinwissen" bestärken, werden z.B. in Elworthy (1994) und Pereira und Schabes (1992) für das EM-Training von Hidden-Markov-Modellen und probabilistischen kontextfreien Grammatiken präsentiert. In diesem Abschnitt wird ein neues stochastisches lexikalisiertes Modell für unifikationsbasierte Grammatiken vorgestellt, für welches der EM-Algorithmus gleich zweimal benutzt wird: zum einen wird eine Instanz des EM-Algorithmus für sogenannte *log-lineare Modelle* (Riezler 1999) benutzt, um das Modell auf freiem Text zu trainieren. Zum anderen basiert dieses Modell auf den vorab estimierten klassenbasierten Häufigkeiten eines EM-basierten Klassifikationsmodells (Prescher et al. 2000).

Die experimentellen Resultate, die in Abschnitt 5.4 vorgestellt werden, werden zeigen, dass in dieser zweimaligen Anwendung des EM-Algorithmus die statistische Modellierung einer unifikationsbasierten Grammatik sehr akurat vorgenommen werden kann.

Log-lineare Modelle

Eine *log-lineare Wahrscheinlichkeitsverteilung* $p_\lambda(x)$ auf der Menge $\mathcal{X}$ der Analysen einer unifikationsbasierten Grammatik ist wie folgt definiert:

$$p_\lambda(x) = {Z_\lambda}^{-1} e^{\lambda \cdot \nu(x)} p_0(x) \ .$$

Hierbei gilt:

- Z_λ ist eine *Normalisierungskonstante*

$$Z_\lambda = \sum_{x \in \mathcal{X}} e^{\lambda \cdot \nu(x)} p_0(x) \ ,$$

- λ ist ein n-dimensionaler *Parametervektor*,

$$\lambda = (\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^n \ ,$$

- ν ist ein n-dimensionaler *Eigenschaftsvektor*

$$\nu = (\nu_1, \ldots, \nu_n) \ ,$$

  wobei eine *Eigenschaft* lediglich als eine Funktion aufgefasst wird, die jeder Analyse eine reelle Zahl[4] zuordnet ($i = 1 \ldots n$):

$$\nu_i : \ \mathcal{X} \rightarrow \mathbb{R} \ ,$$

- $\lambda \cdot \nu(x)$ ist das *Skalarprodukt*

$$\lambda \cdot \nu(x) = \sum_{i=1}^{n} \lambda_i \nu_i(x) \ ,$$

- p_0 ist eine gegebene und fixierte *Referenzverteilung*

$$p_0 : \ \mathcal{X} \rightarrow \mathbb{R} \ .$$

Die Aufgabe der probabilistischen Modellierung mit log-linearen Wahrscheinlichkeitsverteilungen kann als eine zweigeteilte Aufgabe angesehen werden:

(i) Zum einen müssen angemessene Eingenschaften ν_i gefunden werden ($i = 1 \ldots n$), die mit Erfolg in das log-lineare Modell integriert werden können. Dies ist im allgemeinen ein kreativer Akt, der durchaus mit dem Schreiben von Grammatikregeln verglichen werden kann. Es wird sich zeigen, dass hier globale anstatt lokale Eigenschaften gewählt werden sollten, die über die Mächtigkeit von kontextfreien Regeln hinausgehen, um so die volle Mächtigkeit der log-linearen Modelle auszunutzen.

[4] z.B. ihre Vorkommenshäufigkeit in der Analyse, oder den binären Wert 1/0 je nachdem, ob sie oder ob sie nicht in der Analyse vorkommt, etc.

Eingabe: Ein *Trainingskorpus* $\tilde{p}(.)$ von Sätzen $y \in \mathcal{Y}$, eine *symbolische Analyse-Komponente*, die für jeden Satz y seine Parses $X(y)$ angibt, eine *Referenzverteilung* $p_0(.)$ auf den Parses $\mathcal{X}$ des Trainingskorpus, ein *Eigenschaftsvektor* $\nu = (\nu_1, \ldots, \nu_n)$ mit konstantem $\nu_\#(x) = \sum_{i=1}^{n} \nu_i(x)$, die *Anzahl Trainingsiterationen* $numberOfIterations \geq 1$.

Ausgabe: Ein mit dem EM-Algorithmus trainiertes log-lineares Modell p_{λ^*} auf den Parses des Trainingskorpus $\mathcal{X} := \sum_{y \in \mathcal{Y}, \tilde{p}(y)>0} X(y)$.

Prozedur:

1. **initialize** $\lambda := (0 \ldots 0)$;
2. **for each** $iteration = 1 \ldots numberOfIterations$ **do**
3. **compute** $p_\lambda(x)$, $p_\lambda(x|y)$ based on $\lambda = (\lambda_1, \ldots, \lambda_n)$;
4. **for each** $i = 1 \ldots n$ **do**
5. $\gamma_i := \frac{1}{\nu_\#} \log \frac{\sum_{y \in \mathcal{Y}} \tilde{p}(y) \sum_{x \in X(y)} p_\lambda(x|y) \nu_i(x)}{\sum_{x \in \mathcal{X}} p_\lambda(x) \nu_i(x)}$;
6. $\lambda_i := \lambda_i + \gamma_i$;
7. **print** $\lambda^* = (\lambda_1, \ldots, \lambda_n)$;

Abbildung 5.20: EM-Algorithmus für log-lineare Modelle

(ii) Die *statistische Inferenz* oder das *Training* log-linearer Modelle hat die Aufgabe, die Parameter λ so zu estimieren, dass die empirische Wahrscheinlichkeitsverteilung der Trainingsdaten möglichst gut reflektiert wird.

Training log-linearer Modelle mit dem EM-Algorithmus

Für annotierte Daten werden log-lineare Modelle standardmässig mit den *iterativen Skalierungsmethoden* von Darroch und Ratcliff (1972) und Della Pietra et al. (1997) trainiert. Falls keine annotierten Daten zur Verfügung stehen, kann der EM-Algorithmus (Dempster et al. 1977) angewendet werden. Der in Abbildung 5.20 gezeigte Algorithmus ist ein doppelt iterativer Algorithmus, der eine Instanz des EM-Algorithmus für log-lineare Modelle darstellt (Riezler 1999).

Wird dieser Algorithmus auf stochastische unifikationsbasierte Grammatiken angewendet, so wird angenommen, dass die folgenden Elemente gegeben sind:

- Ein Trainingskorpus, bestehend aus freiem Text. Jeder Satz wird mit der empirischen Wahrscheinlichkeit $\tilde{p}(y)$ beobachtet.

- Eine unifikationsbasierte Grammatik, die jedem Satz y die Menge $X(y)$ seiner Analysen zuordnet.
- Ein log-lineares Modell $p_\lambda(\cdot)$ für die Parses $\mathcal{X} = \sum_{y \in \mathcal{Y} | \tilde{p}(y) > 0} X(y)$ der Sätze des Trainingskorpus, mit bekannten Werten für die Eigenschaften des Vektors ν, aber mit unbekannten Werten für die Paramter λ.

Das Ziel der Maximum-Likelihood-Estimierung ist es (siehe Abschnitt 2.4), einen Parameter λ^* zu finden, der die Log-Likelihood:

$$L = \sum_{y \in \mathcal{Y}} \tilde{p}(y) \log \sum_{x \in X(y)} p_\lambda(x)$$

maximiert, d.h.

$$\lambda^* = \text{argmax}_{\lambda \in \mathbb{R}^n} L(\lambda) \ .$$

Der in Abbildung 5.20 gezeigte Algorithmus ist eine Instanz des EM-Algorithmus und setzt voraus,

1. dass $\nu_\#(x) = \sum_{i=1}^n \nu_i(x)$ konstant ist, und
2. dass das log-lineare Modell eine Wahrscheinlichkeitsverteilung auf den Parses des Trainingskorpus ist[5].

Die erste Voraussetzung kann erfüllt werden, indem eine sogenannte *Korrektureigenschaft*

$$\nu_0(x) = K - \nu_\#(x)$$

(mit einem konstanten $K \in \mathbb{R}$) eingeführt wird, da dann

$$\sum_{i=0}^{n} \nu_i(x) = \nu_0(x) + \sum_{i=1}^{n} \nu_i(x) = (K - \nu_\#(x)) + \nu_\#(x) = K$$

für alle $x \in \mathcal{X}$ ist[6].

Wegen der zweiten Voraussetzung, wird das log-lineare Modell im allgemeinen keine Wahrscheinlichkeitsverteilung für die Menge aller möglichen Analysen der unifikationsbasierten Grammatik darstellen. Daher sollte in der Anwendung dieser Modelle, die typischerweise in der Disambiguierung alternativer Analysen bestehen wird, deren Sätze nicht aus dem Trainingskorpus stammen, eher von log-linearen Massen gesprochen werden.

[5] Im Gegensatz hierzu lieferen kontextfreie Grammatiken eine Wahrscheinlichkeitsverteilung auf der ganzen Sprache der Grammatik.

[6] In der Regel wird $K = \max_{x \in \mathcal{X}} \nu_\#(x)$ gewählt, was allerdings nicht zwingend erforderlich ist.

Startparameter für log-lineare Modelle

Wie in Abschnitt 3.6 vorgestellt wurde, erzeugt der EM-Algorithmus eine Folge von Parametern, deren Log-Likelihoodwerte monoton wachsend gegen einen stationären Wert der Log-Likelihood konvergieren. In Abschnitt 3.6 wurde aber ebenfalls vorgestellt, dass der erreichte Log-Likelihoodwert nicht notwendig ein globales Maximum sein wird, sondern in Abhängigkeit des Startparameters ein lokales Minimum, ein lokales Maximum oder ein Sattelpunkt sein wird. In diesem Abschnitt wird eine überraschend einfache Methode vorgestellt, die Ordnung in dieses chaotische Konvergenzverhalten des EM-Algorithmus bringen könnte, indem nach Startparametern gesucht wird, die in der Nähe des globalen Maximums der Log-Likelihood liegen. Die Problemstellung kann am besten für annotierte Daten mit dem sogenannten *Minimum-Divergenz-Paradigma* von Kullback erklärt werden, welches in diesem Fall zur Maximum-Likelihood-Estimierung äquivalent ist:

> Die Wahrscheinlichkeitsverteilung p^*, welche die Divergenz $D(p||p_0)$ zu einer Referenzverteilung p_0 minimiert, und die Nebenbedingungen $p[\nu_i] = \tilde{p}[\nu_i],\ i = 1 \ldots n$ erfüllt, ist dasjenige log-lineare Modell p_{λ^*}, welches die Log-Likelihood $L(\lambda) = \tilde{p}[\log p_\lambda]$ der Trainingsdaten maximiert[7].

Die Wahl von Startwerten

$$\lambda_i = 0 \text{ für } i = 1, \ldots, n$$

ist daher (zumindest für annotierte Daten) sehr vernünftig, da (i) diese Startparameter das Modell $p_\lambda = p_0$ ergeben, welches offensichtlich eine minimale Divergenz zu der Referenzverteilung p_0 aufweist und (ii) nach dem ersten Re-Estimierungsschritt auch die Nebenbedingungen

$$p[\nu_i] = \tilde{p}[\nu_i],\ i = 1 \ldots n$$

erfüllt. Im nächsten Abschnitt wird gezeigt, dass die vorgestellte, theoretisch gut motivierte Initialisierung der Parameter mit Nullwerten, im Vergleich zu zufällig initialisierten Modellen (wie es der EM-Algorithmus im allgemeinsten Fall vorschreibt), sowohl verbesserte Maximum-Likelihood-Estimierungen ergibt, als auch Modelle, die in der linguistischen Evaluierung bessere Ergebnisse aufweisen.

[7] Die relative Entropie oder Kullback-Leibler-Distanz $D(.||.)$ wird gelegentlich auch Divergenz genannt. Es ist interessant, und einfach zu beweisen, dass das Minimum-Divergenz-Modell unter Benutzung einer uniformen Referenzverteilung p_0 zu einem Maximum-Entropie-Modell wird (Jaynes 1957).

5.4 Experimente mit einer deutschen LFG

Die in diesem Abschnitt präsentierten Experimente basieren auf *log-linearen Wahrscheinlichkeitsmodellen* und inkorporierten *EM-basierten Klassifikationsmodellen*. Das kombinierte Modell definiert eine Wahrscheinlichkeitsverteilung auf den Syntaxanalysen einer *LFG-Grammatik* für Deutsch. In der Arbeit von Johnson et al. (1999) wurde mit Hilfe der sogenannten *Pseudo-Likelihood-Estimierung* erstmals mit log-linearen Modellen auf kleineren annotierten Korpora experimentiert, mit dem Ziel, stochastische unifikationsbasierte Grammatiken zu induzieren. Leider stehen bis zum heutigen Zeitpunkt keine mit einer LFG-Grammatik annotierten Korpora von freiem deutschen Text zur Verfügung stehen. In den folgenden Experimenten wurde daher auf das in Abschnitt 5.3 vorgestellte Trainingsverfahren für log-linearen Modelle zurückgegriffen, welches glücklicherweise nicht auf annotierte Trainingsdaten angewiesen ist.

In dem grössten Experiment werden 250 000 Syntaxanalysen von ca. 36 000 Sätzen benutzt, die mit der deutschen LFG geparst wurden. Die experimentelle Evaluierung der Modelle wird mit dem üblichen "Exact-Match-Kriterium" (d.h. exakte Übereinstimmung der wahrscheinlichsten Syntaxanalyse mit der korrekten Syntaxanalyse) eine Präzision von 86% ergeben. Zur Messung dieser Ergebnisse wurde ein Evaluierungskorpus mit 550 manuell annotierten Sätzen mit durchschnittlich 5.4 Syntaxanlysen erstellt. Eine zweite Evaluierung, in der die Subkategorisierungsrahmen von Verben erkannt werden müssen, wird eine Präzision (Übereinstimmung der Subkategorisierungsrahmen des Hauptverbs in wahrscheinlichster und korrekter Syntaxanalyse) von 90% ergeben. Hier wurde ein Evaluierungskorpus mit 375 manuell disambiguierten Beispielen mit einer durchschnittlichen Ambiguität von 25 Syntaxanalysen erstellt. Eine direkte Bewertung dieser Ergebnisse mit anderen statistischen Parsern ist schwierig, da hierzu verschiedene Trainingdaten und Evaluierungsmetriken berücksichtigt werden müssten. Trotzdem werden die folgenden Schlüsse aus dem Experiment gezogen werden:

- Die Lexikalisierung mit klassenbasierten Häufigkeiten verbessert die Präzision um mehr als 10% (gegenüber unlexikalisierten Modellen).

- Für log-lineare Modelle lassen sich für den EM-Algorithmus gut motivierbare und in Experimenten gut bewährte *Startparameter* finden.

- Der EM-Algorithmus verbessert die Estimierung der Wahrscheinlichkeitsparameter von unifikationsbasierten Grammatiken (um mehr als 10% Präzision gegenüber dem Baumbank-Trainingsverfahren).

Log-lineare Modelle: 190 Basis-Eigenschaften

Die 190 Basis-Eigenschaften, die in diesem Experiment benutzt wurden, sind sehr ähnlich zu den Eigenschaften, die Johnson et al. (1999) benutzten, um generelle linguistische Prinzipien in ihr log-lineares Modell einzubetten. Die Eigenschaften beziehen sich zum einen auf die sogenannte c-Struktur (die Konstituentenstruktur), zum anderen auf die f-Struktur (die Featurestruktur) der LFG-Analysen. Beispielsweise wurden die folgenden Eigenschaften verwendet:

- c-Strukturen, die zu kontextfreien Standard-Grammatikregeln korrespondieren,
- c-Struktur-Teilbäume, die Argument- versus Adjunkt-Attachment ausdrücken,
- f-Struktur-Attribute, die zu den in LFG benutzten grammatischen Funktionen korrespondieren,
- atomare Attribut-Wert-Paare in f-Strukturen,
- Bewertungen, ob die c-Struktur rechtsverzweigend ist,
- Bewertungen, ob Koordination in der c-Struktur parallel realisiert wird.

Lexikalisierung

Der hier vorgestellte Ansatz zur Lexikalisierung ist zum einen klassenbasiert, da er estimierte klassenbasierte Häufigkeiten $f_c(v,n)$ von Verbköpfen v und Argumentköpfen n benutzt, zum anderen verallgemeinert und verschmilzt er die Ansätze, die mit rein frequenzbasierten Statistiken oder rein klassenbasierten Wahrscheinlichkeiten von lexikalischen Köpfen in grammatischen Relationen arbeiten. Estimierte klassenbasierte Häufigkeiten werden in Prescher et al. (2000) als Häufigkeiten $f(v,n)$ von (v,n)-Paaren eingeführt, die durch die Wahrscheinlichkeiten $p(c|v,n)$ der "Klassenmitgliedschaft" eines EM-basierten Klassifikationsmodells gewichtet werden. Sind

$$f_c(v,n) = p(c|v,n) \cdot (1 + f(v,n)), \qquad (c \in C)$$

die klassenbasierten Häufigkeiten, so ist

$$f_{\text{class-based}}(v,n) = \max_{c \in C} \; f_c(v,n)$$

die maximale klassenbasierte Häufigkeit. Wie in Prescher et al. (2000) in einer Evaluierung zur Auflösung lexikalischer Ambiguitäten gezeigt wird, kann ein Gewinn von ca. 7% Präzision erzielt werden, indem die klassenbasierten Häufigkeiten $f_{\text{class-based}}(v,n)$ anstelle der klassenbasierten Wahrscheinlichkeiten $p(n|v)$ zur Disambiguierung benutzt werden.

Um diese Beobachtung möglichst gut auszunutzen, wurden die Entscheidungen dieses Disambiguators in 45 weiteren Eigenschaften des log-linearen Modells eingebettet. Hierzu wurden unter anderem die klassenbasierten Häufigkeiten der lexiaklischen Köpfe in den grammatischen Relationen des Subjekts, des direkten, und indirekten Objekts, sowie der Dativ- und Akkusativ-PPs von maximal drei Verben einer jeden Analyse berücksichtigt. Ist $v_r(x)$ der verbale Kopf, bzw. $n_r(x)$ der nominale Kopf in einer grammatischen Relation r des Parses x, so kann eine binäre lexikalisierte Eigenschaft ν_r für die grammatische Relation r definiert werden:

$$\nu_r(x) = \begin{cases} 1 & \text{falls } f_{\text{class-based}}(v_r(x), n_r(x)) \geq f_{\text{class-based}}(v_r(x'), n_r(x')) \; \forall x' \in X(y), \\ 0 & \text{sonst.} \end{cases}$$

Die Eigenschaft ν_r kann dahingehend interpretiert werden, dass sie die Parses $x \in X(y)$ eines Satzes y gemäss der klassenbasierten Häufigkeit prä-disambiguiert und nur die Information, welches der beste Parse ist, nicht aber die aktuellen Häufigkeitswerte speichert. In der Evaluierung wird sich zeigen, dass diese Prädisambiguierungs-Routine die Performanz der Modelle um mehr als 10% steigern wird.

Experimente zum Vergleich von Parsebanken und unannotierten Daten

In den Experimenten wurde eine LFG-Grammatik für Deutsch benutzt[8], um freien Zeitungstext zu parsen. Um die Anzahl der Syntaxanalysen zu kontrollieren wurden die Trainingsdaten auf Sätze mit weniger als 20 Parses beschränkt. Der finale Trainingskorpus bestand aus ca. 36 000 Sätzen und ca. 250 000 Parses.

Hieraus wurde ein Teilkorpus von ca. 4 000 Sätzen extrahiert, für die die LFG eine eindeutige Analyse lieferte. Die durchschnittliche Satzlänge von 7.5 Worten auf dieser automatisch hergestellten "Parse-Bank" ist nur geringfügig kleiner als die Satzlänge von 10.5 Worten auf dem vollen Korpus. Daher kann angenommen werden, dass sich in der Parsebank die meisten linguistischen Phänomene finden lassen, die im vollen Korpus vorkommen. Das Training mit dieser automatisch hergestellten Parse-Bank hat dieselben vorzüglichen Eigenschaften, wie das Training mit einer manuell erstellten Baumbank. Z.B. ist die Konvergenz gegen ein globales Maximum der Log-Likelihood gewährleistet, was möglicherweise keine schlechte Voraussetzung ist, um Estimierungen zu erhalten, die auch in der linguistischen Evaluierung gute Ergebnisse zeigen. Die mit dieser Parse-Bank trainierte Grammatik

[8]Die deutsche LFG-Grammatik wurde in der Grammatik-Entwicklungsumgebung XLE (Maxwell und Kaplan 1996) implementiert. Die Abdeckung der Grammatik liegt bei ca. 50% (für freien Zeitungstext). Für die Disambiguierungsaufgabe spielt die Abdeckung der LFG aber offensichtlich keine entscheidende Rolle.

kann daher zu einem interessanten Vergleich mit der auf dem vollen Korpus trainierten Grammatik herangezogen werden.

Testkorpora und Evaluierungsaufgaben

Um die Modelle zu evaluieren, wurden zwei verschiedene Testkorpora erstellt. Zum einen wurden mit der LFG-Grammatik 550 Beispielsätze aus der deutschen Grammatik von Helbig und Buscha (1996), einem Handbuch für den Ausländerunterricht geparst. Anschliessend wurde der korrekte Parse, gemäss der in Helbig und Buscha (1996) intendierten Lesart, manuell annotiert. Dieses Vorgehen hatte den Vorteil, dass eine präzise Markierung des korrekten c/f-Struktur-Paars möglich war. Leider liegt die durchschnittliche Ambiguität nur bei 5.4 Parses und 7.5 Worten pro Satz. Um auch Sätze mit einer höheren Ambiguitätsrate zu disambiguieren, wurden weitere 375 Sätze aus freiem Zeitungstext manuell annotiert. Die Sätze dieses Korpus haben durchschnittlich 25 Parses und 11.2 Worte.

Die Modelle wurden mit Hilfe von zwei Evaluierungsaufgaben getestet, einer "Exact-Match"- und einer "Frame-Match"-Aufgabe. In der ersten Aufgabe wurde die volle Übereinstimmung des c/f-Struktur-Paars des korrekten Parses mit dem annotierten Parse überprüft. Bei der zweiten Aufgabe wurde lediglich getestet, ob der Subkategorisierungsrahmen des Hauptverbes korrekt erkannt wurde. Offensichtlich ist die Ambiguitätsrate in der zweiten Aufgabe kleiner als in der ersten Aufgabe, und kann daher als die Evaluierung des kompletten Systems (bestehend aus der symbolischen LFG-Grammatik und der statistischen Disambiguierungskomponente) angesehen werden.

Die Performanz wurde für beide Evaluierungsaufgaben gemäss der folgenden Evaluierungsmasse bestimmt:

$$\text{Precision} = \frac{\#\text{correct}}{\#\text{correct}+\#\text{incorrect}},$$
$$\text{Effectiveness} = \frac{\#\text{correct}}{\#\text{correct}+\#\text{incorrect}+\#\text{don't know}}.$$

"Correct" und "incorrect" wird durch Erfolg und Misserfolg auf der Evaluierungsaufgabe spezifiziert; "don't know"-Fälle treten auf, wenn das System nicht in der Lage ist, eine Entscheidung zu fällen, also in Fällen mit mehr als einen wahrscheinlichstem Parse.

Experimentelle Resultate

Für jede Evaluierungsaufgabe und jeden Evaluierungskorpus wurde eine Zufalls-Baseline bestimmt, indem mehrere Modelle mit zufälligen Parameterwerten initialisiert, evaluiert

exact match evaluation	basic model	lexicalized model	selected + lexicalized model
complete-data estimation	P: 68 E: 59.6	P: 73.9 E: 71.6	P: 74.3 E: 71.8
incomplete-data estimation	P: 73 E: 65.4	P: 86 E: 85.2	P: 86.1 E: 85.4

frame match evaluation	basic model	lexicalized model	selected + lexicalized model
complete-data estimation	P: 80.6 E: 70.4	P: 82.7 E: 76.4	P: 83.4 E: 76
incomplete-data estimation	P: 84.5 E: 73.1	P: 88.5 E: 84.9	P: 90 E: 86.3

Abbildung 5.21: Evaluierung: Exact-Match für 550 Sätze mit 5.4 Analysen (oben) und Subkatrahmen-Erkennung für 375 Sätze mit 25 Analysen (unten)

und gemittelt wurden. Diese Baseline misst daher die Disambiguierungsergebnisse des rein symbolischen Parsers.

Die Resultate der Exact-Match-Evaluierung auf dem Helbig-Buscha-Korpus werden in Abbildung 5.21 (oben) gezeigt. Die Spalten zeigen drei Modelle mit jeweils verschiedenen Eigenschaftsvektoren. Die Spalte "basic model" referiert auf die Modelle mit den 190 (oben vorgestellten) Basis-Eigenschaften, während "lexicalized model" die Modelle mit den zusätzlichen 45 lexikalisierten Eigenschaften bezeichnet. Die Spalte "selected + lexicalized model" weist auf lexikalisierte Modelle hin, deren Eigenschaften mit einem einfachen Kriterium selektiert wurden, indem nur solche Eigenschaften berücksichtigt wurden, die mit einer gewissen Häufigkeit im Korpus vorkommen.

Die Zufalls-Baseline liegt bei diesem Experiment bei nur ca. 33%, während die besten, trainierten Modelle eine Performanz von bis zu 86% Präzision (P) aufweisen. Der genaue Vergleich zeigt, dass das Training auf dem vollen Korpus gegenüber dem Training auf der Parse-Bank je nach Modellfamilie einen Gewinn von 5%/ 12.1%/ 11.8% Präzision (P) und 5.8%/ 13.6%/ 13.6% Effektivität (E) ergibt. Die Lexikalisierung zieht je nach Trainingskorpus (Parse-Bank/ voller Korpus) einen noch grösseren Gewinn von 15.9%/ 13% Präzision (P) und 12%/ 19.8% Effektivität (E) nach sich. Der Gewinn durch die

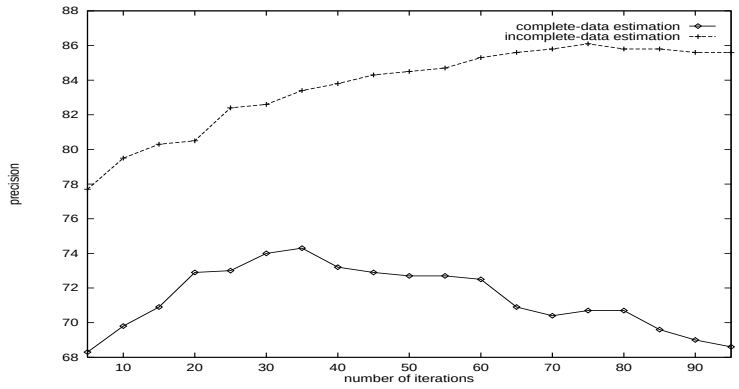


Abbildung 5.22: Precision-Werte für die einzelnen Trainingsiterationen

Selektion von Eigenschaften ist dagegen vernachlässigbar.

Abbildung 5.21 (unten) zeigt, dass der Performanzgewinn sowohl durch Lexikalisierung als auch durch EM-Training kleiner ausfällt, wenn die leichtere Subkatrahmen-Evaluierung durchgeführt wird. Die Zufalls-Baseline liegt hier bei 70% erkannten Subkategorisierungsrahmen auf Sätzen aus freiem Zeitungstext mit durchschnittlich 25 Analysen pro Satz. Die besten Modelle zeigen dagegen eine Präzision von ca. 90% (P). Ein Vergleich zeigt, dass das Training auf dem vollen Korpus gegenüber dem Training auf der Parse-Bank je nach Modellfamilie einen Gewinn von 3.9%/ 5.8%/ 6.6% Präzision (P) und 2.7%/ 8.5%/ 10.3% Effektivität (E) ergibt. Die Lexikalisierung zieht je nach Trainingskorpus (Parse-Bank/ voller Korpus) einen Gewinn von 2.1%/ 4.0% Präzision (P) und 6.0%/ 11.8% Effektivität (E) nach sich. Der Gewinn durch die Selektion von Eigenschaften ist erneut vernachlässigbar.

Die optimale Anzahl von Iterationen wurde bestimmt, indem die Modelle nach jeder 5ten Iteration evaluiert wurden. Abbildung 5.22 zeigt die Precision-Werte der lexikalisierten und selektierten Modelle auf dem Evaluierungskorpus der Exact-Match-Aufgabe. Für das Training mit der Parse-Bank werden maximale Precision-Werte nach 35 Iterationen erreicht[9]. Weitere Iterationen zeigen einen deutlichen Übertrainingseffekt. Für EM-Training sind deutlich mehr Iterationen erforderlich, um die besten Precision-Werte zu erreichen. Ein Vergleich von zufälligen mit uniformen Startwerten zeigt für die letzteren einen Gewinn von 10-40% Präzision. Dies bestätigt die theoretische Annahme, dass uniforme Startwerte die Log-Likelihood in die Nähe ihres finalen Wertes drücken können.

[9]Im Gegensatz zu kontextfreien Grammatiken, die mit Hilfe von Baumbanken in nur einer einzigen Iteration trainiert werden (siehe Abschnitt 2.5), erfordert das Training mit unifikationsbasierten Baumbanken eine iterative Prozedur (siehe Abbildung 5.20 mit $p(x|y) = 1$).

Zusammenfassung

Es wurde ein neuer Ansatz für die stochastische Modellierung unifikationsbasierter Grammatiken präsentiert, der zeigt, dass ein EM-Training erfolgreich eingesetzt werden kann, um akurate stochastische Modelle zu generieren.

Besonders interessant ist, dass für log-lineare Modelle Startparameter gefunden werden konnten, die theoretisch gut motiviert sind, und empirisch gute Ergebnisse erzielen.

Ferner wurde eine Lexikalisierung auf Grundlage klassenbasierter Häufigkeiten vorgestellt, die die Disambiguierungs-Performanz der trainierten Modelle eindrucksvoll verbessern konnte.

Da die hier vorgestellten Modelle mit einer nicht-dynamischen Instanz des EM-Algorithmus trainiert wurden, die es leider erforderlich machte, die durchschnittliche Anzahl der Satzanalysen stark zu beschränken, besteht die zentrale zukünftige Aufgabe darin, Techniken zu entwickeln, die diesen Mangel beheben werden.

In diesem Zusammenhang sollen die erzielten Ergebnisse auch für Evaluierungskorpora bestätigt werden, die weniger stark kontrollierte Sätze mit wesentlich mehr als 5 bis 25 Analysen enthalten.

Kapitel 6

Zusammenfassung

Kapitel 1 führte in die vorliegende Dissertation ein, die sich mit dem Erwartungswert-Maximierungs-Algorithmus (EM-Algorithmus) in Theorie und Praxis beschäftigt.

In Kapitel 2 wurde eine Einführung in die stochastischen und informationstheoretischen Grundlagen der Linguistik gegeben. Hierbei wurde versucht, die behandelten mathematischen Sachverhalte nicht nur linguistisch zu interpretieren, sondern, wenn möglich, auch mit neuen und interessanten Beweisen anzureichern. Zunächst wurden in Kapitel 2 die nötigen stochastischen Begriffe vorgestellt und linguistisch motiviert. Ein einfaches Resultat war beispielsweise, dass jeder linguistische Korpus aus stochastischer Sicht mit der *empirischen Wahrscheinlichkeitsverteilung* der Datentypen im Korpus identifizierbar ist. Anschliessend wurden die *probabilistischen kontextfreien Grammatiken* eingeführt. Es wurde vorgestellt, dass die *Grammatikregel-Wahrscheinlichkeiten* die sogenannte *Standard-Nebenbedingung* erfüllen sollten, um eine der hinreichenden Bedingungen für *Konsistenz* zu gewährleisten. Unter Benutzung eines Theorems von Booth und Thompson (1973) wurde gezeigt, dass die speziellen probabilistischen kontextfreien Grammatiken, die die *EM-basierten Klassifikationsmodelle* aus Kapitel 4 modellieren, konsistent sind. Nach Einführung des zentralen Begriffs des *Erwartungswerts* wurde demonstriert, dass sich die *Korpuswahrscheinlichkeit* in die Erwartungswerte *Log-Likelihood*, *Cross-Entropie*, *Kullback-Leibler-Distanz*, sowie *Korpusperplexität* umrechnen lässt. Anschliessend wurde die fundamentale *Informationsungleichung der Informationstheorie* vorgestellt, die besagt, dass die *relative Entropie* nicht-negativ ist, bzw. dass die Cross-Entropie mindestens so gross ist, wie die *Entropie*. Ein neuer Beweis der wichtigen Beobachtung, dass die empirische Wahrscheinlichkeitsverteilung die Korpuswahrscheinlichkeit maximiert, konnte zu einem neuen Beweis der Informationsungleichung der Informationstheorie ausgebaut werden. Ein weiterer neuer Beweis zeigte, dass die Wahrscheinlichkeit einer *Baumbank* von der *Baumbank-Grammatik* maximiert wird. Der Schlüsselschritt war, dass die *Baumbank-Wahrschein-*

lichkeit ein Wahrscheinlichkeitsprodukt gewisser *Teilkorpora* ist, von denen bekannt ist, dass sie durch ihre jeweiligen empirischen Wahrscheinlichkeitsverteilungen maximiert werden. Ferner wurden in diesem Kapitel die wichtigsten *symbolischen* und *stochastischen Parsingverfahren* für kontextfreie Grammatiken vorgestellt, und gezeigt, dass sie die bekannte Technik der *dynamischen Programmierung* verwenden, und daher durch den *Semiring-Parsing-Algorithmus* (Goodman 1998) verallgemeinert werden können. Schliesslich wurden die drei wichtigsten *Sprachmodelle* eingeführt: *n-gram-Modelle*, *Markov-Modelle* und *Hidden-Markov-Modelle*. Als Hauptergebnis von Kapitel 2 wurde demonstriert, dass alle Sprachmodelle durch probabilistische kontextfreie Grammatiken modelliert werden können.

Im Kapitel 3 wurde zunächst der *Inside-Outside-Algorithmus* vorgestellt, mit dem es möglich ist, eine gegebene kontextfreie Grammatik auf freiem Text, d.h. ohne Hilfe einer manuell erstellten Baumbank, iterativ zu trainieren. Die Re-Estimierungsformeln des Inside-Outside-Algorithmus wurden, dem historischen Ablauf folgend, zunächst für einen Trainingskorpus, bestehend aus einem einzigen Satz (Baker 1979) und anschliessend für beliebig grosse Trainingskorpora (Lari und Young 1990) vorgestellt. Ferner wurde gezeigt, wie die freien Parameter des Inside-Outside-Algorithmus (Grösse des Trainingskorpus, Iterationsanzahl, Startgrammatiken) in einer *systematischen linguistischen Evaluierung* optimal festgelegt werden können. Im Gegensatz zu vielen falschen Zitaten, wurde festgestellt, dass weder Baker (1979) noch Lari und Young (1990) formal bewiesen haben, dass die mit dem Inside-Outside-Algorithmus iterativ trainierten probabilistischen Grammatiken mononton wachsende Log-Likelihoodwerte aufweisen, und gegen ein lokales Maximum der Log-Likelihood-Funktion konvergieren[1]. Anschliessend wurde der EM-Algorithmus anhand seiner symbolischen und stochastischen Komponenten eingeführt und aus prozeduraler Sicht dargestellt. Hierbei wurde eine Version des EM-Algorithmus mit expliziter Variation der Startparameter vorgestellt, da dieser freie Parameter des EM-Algorithmus erfahrungsgemäss den grössten Einfluss auf die Ergebnisse hat. Anschliessend wurden zwei alternative Implementierungen des EM-Algorithmus vorgestellt, die eine unveränderte Funktionalität aufweisen, aber tiefere Einblicke in dessen Funktionsweise ermöglichen und es gestatteten, den EM-Algorithmus in den Kontext des *stochastischen Bootstrap-Algorithmus (SB-Algorithmus)* zu stellen. Der SB-Algorithmus wurde in dieser Dissertation konzipiert, um ein beliebiges *stochastisches Trainingsverfahren auf vollständigen Datentypen* in ein *iteratives stochastisches Trainingsverfahren auf unvollständigen Datentypen* zu transformieren. Der Vorteil des SB-Algorithmus gegenüber der direkten Benuzung des zugrundeliegenden überwachten Trainingsverfahrens liegt darin, dass es mit dem SB-Algorithmus un-

[1] Konvergenz gegen ein lokales Maximum der Log-Likelihood-Funktion wird sowieso nur unter besonderen Voraussetzungen eintreffen.

problematisch ist, *beliebig viele Modellvarianten* (Startparameter, Anzahl Iterationen) auf *ausreichend grossen Trainingskorpora* zu trainieren. Die Annahme ist, dass wenigstens eines dieser Modelle bessere Eigenschaften besitzt, als das mit dem überwachten Trainingsverfahren erstellte Modell. Der Nachteil des SB-Algorithmus liegt darin, dass er eine (zumindest einfache) *symbolische Analysekomponente* voraussetzt. Es wurde gezeigt, dass der EM-Algorithmus eine besonders gut motivierbare Instanz des SB-Algorithmus ist, die man erhält, indem im SB-Algorithmus die Maximum-Likelihood-Estimierung auf vollständigen Datentypen, als zugrundeliegendes überwachtes Trainingsverfahren, benutzt wird. Anschliessend wurde das praktische Training mit dem EM-Algorithmus vorgestellt und vorgeschlagen, wie die trainierten Modelle evaluiert werden können. Kern des Vorschlags ist eine zweimalige *Aufteilung* und teilweise *Annotation* des gegebenen Korpus, die in drei Korpora resultiert: einen *Trainingskorpus (unvollständiger Datentypen)*, einen *Entwicklungs-Testkorpus (vollständiger Datentypen)*, sowie einen *Testkorpus (vollständiger Datentypen)*. Ferner wurden in Kapitel 3 die grundlegenden Eigenschaften des EM-Algorithmus vorgestellt. Dabei wurde schrittweise auf die Beantwortung der Frage hingearbeitet, ob die vom EM-Algorithmus generierten Modelle in wohldefinierter Weise gegen ein Modell streben, dass eine *Maximum-Likelihood-Estimierung auf dem gegebenen Korpus der unvollständigen Datentypen* darstellt. Der originalen Arbeit von Dempster et al. (1977) folgend, wurde der sogenannte *generalisierte EM-Algorithmus (GEM-Algorithmus)* vorgestellt und gezeigt, dass ein GEM-Algorithmus (unter sehr milden Voraussetzungen) eine Folge von Modellen mit monoton wachsender Log-Likelihood erzeugt (Theorem 1), deren Grenzwert zumindest ein *stationärer Punkt* (lokales Maximum, lokales Minimum, oder Sattelpunkt) der Log-Likelihood ist, sobald vorausgesetzt werden darf, dass der Grenzwert existiert und alle Modelle stationäre Punkte der EM-Hilfsfunktion sind (Theorem 4). Verbesserte Konvergenzaussagen zum EM-Algorithmus stammen von Wu (1983), der (ebenfalls unter Annahme milder Voraussetzungen) zeigte, dass an jedem *Häufigkeitspunkt* einer von einem GEM-Algorithmus ausgegebenen Parameterfolge ein *lokales Maximum der Log-Likelihood* vorliegt, wenn (unter anderem) vorausgesetzt werden kann, dass die Log-Likelihood für alle Glieder der Parameterfolge, die keine lokalen Maxima sind, *streng monoton wächst* (Theorem 1). Unter nur geringfügig schärferen Voraussetzungen kann gezeigt werden, dass *Parameter-Konvergenz gegen ein lokales Maximum der Log-Likelihood* vorliegt (Theorem 5). Einige eigene Ergebnissen zum EM-Algorithmus zeigten, dass der in einem GEM-Algorithmus re-estimierte Parameter einer Maximum-Likelihood-Estimierung *parameterinvariant* ist, eine Eigenschaft, mit der sich sowohl Dempster et al. (1977), als auch Wu (1983) beschäftigt haben. Anschliessend wurden die ersten und zweiten partiellen Ableitungen der Log-Likelihood und der EM-Hilfsfunktion vorgestellt. Da beide Funktionen *logarithmische Terme* enthalten, wurde der *Delta-Operator* eingeführt, mit dessen Hilfe

alle nötigen analytischen Operationen gemäss den *Rechenregeln des Delta-Operators* stark vereinfacht werden konnten. Überraschenderweise zeigte sich, dass sich die ersten partiellen Ableitungen der Log-Likelihood (und der EM-Hilfsfunktion) als Erwartungswerte der empirischen Wahrscheinlichkeitsverteilung für vollständige Datentypen darstellen lassen. Ferner konnte gezeigt werden, dass eine konvergente Folge von EM-Parametern ein lokales Maximum der Log-Likelihood als Grenzwert besitzt, sobald die *erwartete Cross-Entropie aller Analysen* in einem bestimmten Sinn "kontrollierbar" ist. Anschliessend wurde der EM-Algorithmus auf probabilistische kontextfreie Grammatiken angewendet. Neben den Re-Estimierungsformeln wurden die Konvergenzeigenschaften dieser Instanz des EM-Algorithmus vorgestellt. Es zeigte sich, dass die Log-Likelihood während des Trainings monoton wächst, und die Folge der re-estimierten Parameter gegen einen *stationären Punkt* der Log-Likelihood konvergiert. Falls zusätzlich vorausgesetzt werden kann, dass alle Mutterkategorien während der gesamten Re-Estimierung eine positive Vorkommenserwartung haben und die *erwartete Cross-Entropie aller Analysen* kontrollierbar ist, so liegt sogar Konvergenz gegen ein *lokales Maximum* der Log-Likelihood vor. Somit konnte gezeigt werden, dass die Verhältnisse sehr viel komplexer sind, als in der Literatur gemeinhin angenommen wurde. Ferner wurde gezeigt, dass das *Baumbank-Trainingsverfahren* ein Spezialfall des EM-Algorithmus für annotierte Datentypen und probabilistische kontextfreie Grammatiken ist. Hierfür wurde der *EM-Algorithmus für annotierte Datentypen* eingeführt und gezeigt, dass dieser Algorithmus den optimalen Parameter stets nach einer einzigen Iteration findet. Angewandt auf eine Baumbank lieferte er die bekannten Re-Estimierungsformeln, sowie einen sehr kurzen Beweis der ausgezeichneten Eigenschaft, dass die so estimierten Grammatikregel-Wahrscheinlichkeiten die Baumbank-Wahrscheinlichkeit global maximieren. Anschliessend wurden die formalen Beweise der allseits bekannten Aussagen über den Inside-Outside-Algorithmus, die man entgegen der in der Linguistik weitverbreiteten Meinung (Manning und Schütze 1999) nicht in der Original-Literatur ((Baker 1979), (Lari und Young 1990)) und nach bestem Wissen des Autors auch nirgendwo sonst findet, nachgeholt. Hierbei handelt es sich um die Beweise für die folgenden Aussagen:

- Die Log-Likelihood des Inside-Outside-Algorithmus wächst während des Trainings mononton,
- Die Werte der Log-Likelihood des Inside-Outside-Algorithmus konvergieren gegen einen *stationären Wert*, in einigen Fällen sogar gegen ein *lokales Maximum*.

Für den Beweis dieser beiden Eigenschaften wurde der Weg beschritten, die folgende Aussage formal zu beweisen:

- Der Inside-Outside-Algorithmus ist eine *dynamische Programmiervariante* des EM-Algorithmus für kontextfreie Grammatiken.

Auch diese Eigenschaft des Inside-Outside-Algorithmus ist in der Linguistik relativ gut bekannt, aber (nach bestem Wissen des Autors) ebenfalls noch nie formal bewiesen worden. Zudem stellte sich heraus, dass der exakte Beweis sehr viel weniger leicht ist, als üblicherweise angenommen wurde (Manning und Schütze 1999) und relativ tiefe Kenntnisse über den Inside-Outside-Algorithmus und den EM-Algorithmus erfordert. Der Schlüsselschritt des Beweises war, dass die *Grammatikregel-Counts* aus dem Inside-Outside-Algorithmus formal mit den *erwarteten Grammatikregel-Vorkommenshäufigkeiten* (auf der Menge der Syntaxbäume eines Satzes) im EM-Algorithmus identifiziert werden konnten. Hierzu wurden zum einen die Rekursionsformeln der Inside- und Outside-Wahrscheinlichkeiten benutzt, zum anderen die Rechenregeln des Erwartungswerts.

In Kapitel 4 wurden zunächst die theoretischen Grundlagen des EM-basierten Klassifikationsverfahrens für mehrdimensionale Datentypen vorgestellt und gezeigt, dass das EM-basierte Klassifikationsverfahren tatsächlich eine Instanz des EM-Algorithmus ist. Anschliessend wurden die formalen Eigenschaften des EM-basierten Klassifikationsverfahrens mit Blick auf drei Ziele untersucht: (i) es wurden die *Konvergenzeigenschaften* des EM--basierten Klassifikationsverfahrens betrachtet, wobei unter anderem der Beweis für die *Re-Estimierungsformeln* nachgeholt wurde, (ii) die *Glättungseigenschaften* der mit dem EM-basierten Klassifikationsverfahren trainierten Modelle wurden theoretisch und experimentell überprüft und (iii) es wurde eine *Pruning-Technik* vorgestellt, die dazu führt, dass die mit dem EM-basierten Klassifikationsverfahren trainierten Modelle nur wenig mehr Speicherplatz benötigen, wie die originalen Korpora auf denen sie trainiert wurden. Daran anschliessend wurde eine zweimalige Anwendung des EM-Algorithmus präsentiert: (i) eine Methode zur Konstruktion von Klassifikationsmodellen für Verb-Nomen-Paare, sowie (ii) eine Methode zur maschinellen *Induktion von semantisch annotierten Subkategorisierungsrahmen*. Die Algorithmen und ihre Implementierungen waren effizient genug, um einen Korpus von ca. 100 Millionen Wörtern in ein Lexikon zu übertragen. Die Methode ist für jede natürliche Sprache einsetzbar, für die es *Texte genügender Grösse*, sowie ein *maschinelles Morphologiesystem* und einen *robusten Parser* gibt, der fähig ist, Subkategorisierungsrahmen mit lexikalischen Köpfen zu extrahieren. Anschliessend wurde ein maschinelles Verfahren zur *Identifikation von Verb-Nomen-Kollokationen in Textkorpora* vorgestellt, das erneut als Schlüsselschritt ein EM-basiertes Klassifikationsmodell benutzte. Da sich Kollokationen gerade dadurch auszeichen, dass (i) die beteiligten Lexeme nur selten auch mit anderen Partnern kombinieren und (ii) die Semantik der Kollokation sich nicht aus der Semantik der beteiligten Lexeme erschliessen lässt, kann das vorgestellte,

neue Verfahren als Ergänzung des vorgestellten Verfahrens zum Aufbau eines semantisch annotierten Wörterbuchs aufgefasst werden. Anschliessend wurde gezeigt, dass durch ein *einfaches Nachschlagen in einem semantisch annotierten Lexikon* sehr gute Disambiguierungsresultate erzielbar sind. Jeder Eintrag eines solchen Lexikons besteht im wesentlichen aus einer Wahrscheinlichkeitsverteilung über semantische Klassen, mit denen die Subjekt- und Objektpositionen intransitiver und transitiver Verben annotiert werden. Die Disambiguierung möglicher Zielworte konnte mit diesem Lexikon auf sehr naheliegende Weise vorgenommen werden, indem im Lexikon nach demjenigen Nomen (unter den möglichen Zielworten) gesucht wurde, dessen Semantik am besten mit der Semantik des Zielverbs übereinstimmt. Diese einfache Methode wurde auf einer grossen Anzahl von Übersetzungen eines natürlichen bilingualen Korpus evaluiert. Es zeigte sich, dass sie ähnlich gute Ergebnisse liefert, wie die von Dagan und Itai (1994), in der aber wesentlich mehr Selektionsinformationen benutzt wird. Abschliessend wurden *probabilistische Silbenmodelle* für Englisch vorgestellt, die mit dem mehrdimensionalen EM-basierten Klassifikationsverfahren auf unannotierten drei- und fünfdimensionalen Daten trainiert wurden. Die erhaltenen Silbenmodelle wurden mit einem *Pseudo-Disambiguierungsverfahren* evaluiert. Das Ergebnis zeigte, dass der *Onset* der variabelste Teil der Silbe ist. Die qualitative linguistische Evaluierung zeigte, dass (ohne dass dies vorab zu erwarten war) die Silbenmodelle tatsächlich bedeutungstragende Klassen besitzen.

In Kapitel 5 wurden *probabilistische lexikalisierte kontextfreie Grammatiken* vorgestellt. Zunächst wurde mit drei Beispielen (Briscoe und Carroll 1993) motiviert, dass diese Grammatiken einige natürlichsprachige Phänomene besser modellieren können, als gewöhnliche probabilistische kontextfreie Grammatiken. Dann wurden *lexikalisierte Syntaxbäume* und *lexikalisierte kontextfreie Grammatiken* vorgestellt. Deren Grammatikregeln sind durch *Kopfmarkierungen* angereichert, um *lexikalische Köpfe* verwalten zu können. Das Wahrscheinlichkeitsmodell der lexikalisierten kontextfreien Grammatiken basiert auf *lexikalisch annotierten Grammatikregeln* und sogenannten *lexikalischen Entscheidungen*. Ein wichtiges Ziel von Kapitel 5 war, anhand eines umfangreichen Beispiels vorzustellen, dass sich jede probabilistische lexikalisierte kontextfreie Grammatik durch eine *Grammatiktransformation* auf eine gewöhnliche probabilistische kontextfreie Grammatik zurückführen lässt (Carroll und Rooth 1998). Dies bedeutet, dass der Inside-Outside-Algorithmus bzw. der EM-Algorithmus prinzipiell auch auf lexikalisierte kontextfreie Grammatiken anwendbar ist. Abschliessend wurde vorgestellt, welche Nebenbedingungen die Wahrscheinlichkeitsparamter der lexikalisierten Grammatik erfüllen müssen, um die Standard-Nebenbedingung für kontextfreie Grammatiken nach Booth und Thompson (1973) zu erfüllen. Ein Experiment für das Deutsche, in dem auf einem Korpus von ca 450 000 Sätzen eine lexikalisierte kontextfreie Grammatik trainiert wird, zeigte, dass lexikalisierte kontext-

freie Grammatiken gegenüber unlexikalisierten Grammatiken zumindest gering verbesserte Evaluierungsergebnisse aufweisen. Hierfür wurden nicht die beim Baumbank-Trainingsverfahren üblichen "Exact-Match"-Evaluierungen vorgenommen, sondern sorgfältige linguistische Evaluierungen. Mit manuell erstellten, grossen Evaluierungskorpora wurde satzweise gemessen, wie gut Nominalphrasen und verschiedene Typen von Subkategorisierungsrahmen von den Grammatiken erkannt werden können. Ein weiteres Ergebnis des Experiments war, dass zu viele (mehr als zwei) Iterationen mit dem Inside-Outside-Algorithmus die Ergebnisse der linguistischen Evaluierung verschlechterten. Das Kriterium, das im allgemeinen benutzt wird, um Übertraining zu erkennen, nämlich die Messung der Korpuswahrscheinlichkeit auf einem *Held-Out-Korpus*, zeigte, dass sehr viel mehr Trainings-Iterationen vorgenommen werden müssten. Dies widerspricht der linguistischen Evaluierung, somit kann das Standard-Kriterium als unangemessen gelten. Anschliessend wurde ein neues stochastisches lexikalisiertes Modell für unifikationsbasierte Grammatiken vorgestellt, für welches der EM-Algorithmus gleich zweimal benutzt wurde: zum einen wurde eine Instanz des EM-Algorithmus für sogenannte *log-lineare Modelle* benutzt, um das Modell auf freiem Text zu trainieren. Zum anderen basierte dieses Modell auf den vorab estimierten klassenbasierten Häufigkeiten eines EM-basierten Klassifikationsmodells für Verb-Argument-Paare. Es wurde vorgestellt, wie die Parameter der log-linearen Modelle so estimiert werden können, dass die empirische Wahrscheinlichkeitsverteilung der Trainingsdaten möglichst gut reflektiert wird. Ein wichtiges Ziel war ferner, theoretisch zu begründen, dass für log-lineare Modelle "optimale" *Startparameter* existieren. Dieses Ergebnis ist deshalb so wichtig, weil der allgemeine EM-Algorithmus im Gegensatz zu diesem Ergebnis anweist, möglichst viele Startparameter auszuprobieren. Anschliessend wurde ein Experiment für Deutsch präsentiert, in dem ein Korpus von ca. 36 000 Sätzen mit einer LFG-Grammatik geparst wurde. Es wurden folgende Schlüsse aus dem Experiment gezogen: (i) Die in Abschnitt 5.3 motivierten Startwerte bewähren sich in der Praxis hervorragend. (ii) Die Lexikalisierung mit klassenbasierten Häufigkeiten verbessert die Präzision um ca. 10% (gegenüber unlexikalisierten Modellen). (iii) Der EM-Algorithmus verbessert die Estimierung der Wahrscheinlichkeitsparameter von unifikationsbasierten Grammatiken um ca. 10% Präzision (gegenüber einem Baumbank-Trainingsverfahren). Hierzu wurden 190 überwiegend *nicht-lokale Basis-Eigenschaften* vorgestellt, sowie 45 weitere Eigenschaften, die die *Lexikalisierung des stochastischen Modells* ausmachen. Die experimentelle Evaluierung der Modelle mit dem üblichen "Exact-Match-Kriterium" (d.h. exakte Übereinstimmung der wahrscheinlichsten Syntaxanalyse mit der korrekten Syntaxanalyse) ergaben eine Präzision von 86%. Zur Messung dieser Ergebnisse wurde ein Evaluierungskorpus mit 550 manuell annotierten Sätzen mit durchschnittlich 5.4 Syntaxanlysen erstellt. Eine zweite Evaluierung, in der die Subkategorisierungsrahmen von

Verben erkannt werden müssen, ergab eine Präzision (Übereinstimmung der Subkategorisierungsrahmen des Hauptverbs in wahrscheinlichster und korrekter Syntaxanalyse) von 90%. Hier wurde ein Evaluierungskorpus mit 375 manuell disambiguierten Beispielen mit einer durchschnittlichen Ambiguität von 25 Syntaxanalysen erstellt.

Literatur

Abney, S. (1996). Chunk stylebook. Technical report, SfS, Universität Tübingen.

Abney, S. und M. Light (1999). Hiding a semantic hierarchy in a markov model. In *Proceedings of the ACL'99 Workshop on Unsupervised Learning in Natural Language Processing*, College Park, MA.

Aho, A. V. und J. D. Ullman (1972). *The Theory of Parsing, Translation and Compiling*, Volume I: Parsing. NJ: Prentice-Hall.

Aigner, M. (1996). *Diskrete Mathematik* (2 ed.). Vieweg.

Baayen, H. R., R. Piepenbrock, und H. van Rijn (1993). The CELEX lexical database—Dutch, English, German. *(Release 1)[CD-ROM]. Philadelphia, PA: Linguistic Data Consortium, Univ. Pennsylvania.*

Baker, J. K. (1979). Trainable grammars for speech recognition. In D. Klatt und J. Wolf (Eds.), *Speech Communication Papers for ASA'97*, pp. 547–550.

Bauer (1978). *Wahrscheinlichkeitstheorie und Grundzüge der Masstheorie*. Springer.

Baum, L. E. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities III*, 1–8.

Baum, L. E. und J. A. Eagon (1966, November). An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. Communicated by R. C. Buck.

Baum, L. E., T. Petrie, G. Soules, und N. Weiss (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Math. Statistics 41*(1), 164–171.

Baum, L. E. und G. A. Sell (1968). Growth transformations for functions on manifolds. *Pacific Journal of Mathematics 27*(2), 211–227.

Beil, F., G. Carroll, D. Prescher, S. Riezler, und M. Rooth (1998). Inside-outside estimation of a lexicalized PCFG for German. –Gold–. In *Inducing Lexicons with the EM Algorithm*, AIMS Report 4(3). IMS, Universität Stuttgart.

Beil, F., G. Carroll, D. Prescher, S. Riezler, und M. Rooth (1999). Inside-outside estimation of a lexicalized PCFG for German. In *Proceedings of ACL'99*, College Park, MD.

Benedi, J. und J.-A. Sanchez (2000). Combination of n-grams and stochastic context-free grammars for language modeling. In *Proceedings of COLING'00*, Saarbrücken.

Berger, A. und J. Lafferty (1999). Information retrieval as statistical translation. In *Proceedings of the 22nd annual conference on research and development in information retrieval (SIGIR'99)*.

Black, E. (1992). Meeting of interest group on evaluation of broad-coverage grammars of English. LINGUIST List 3.587, http://www.linguistlist.org/issues/3/3-587.html.

Bod, R. (2000). A probabilistic corpus-driven model for lexical-functional analysis. In *Proceedings of the 18th international conference on computational linguistics (COLING 2000)*, Saarbrücken.

Bod, R. und R. Kaplan (1998). A probabilistic corpus-driven model for lexical-functional analysis. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (COLING/ACL 98)*, Montreal.

Booth, T. L. und R. A. Thompson (1973). Applying probability measures to abstract languages. *IEEE Transactions on Computers C-22*(5), 442–450.

Bosch, K. (1995). *Elementare Einführung in die Wahrscheinlichkeitsrechnung* (6 ed.). Vieweg.

Bosch, K. (1997). *Elementare Einführung in die angewandte Statistik* (6 ed.). Vieweg.

Boyles, R. A. (1983). On the convergence of the *em* algorithm. *J.R.Statist.Soc. 45*(1), 47–50.

Briscoe, T. und J. Carroll (1993). Generalized probabilistic LR parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics 19*(1), 25–59.

Brown, P., P. deSouza, R. Mercer, V. D. Pietra, und J. Lai (1992). Class-based n-gram models of natural language. *Computational Linguistics 18*(4), 467–479.

Brown, P. F., J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, und P. S. Roosin (1990). A statistical approach to machine translation. In *Computaional Linguistics, 16(2)*.

Carroll, G. (1995). *Learning Probabilistic Grammars for Language Modeling*. Ph. D. thesis, Department of Computer Science, Brown University.

Carroll, G. (1997a). *Manual pages for* `charge`, `hyparCharge`. IMS, Universität Stuttgart.

Carroll, G. (1997b). *Manual pages for* `supar`, `ultra`, `hypar`. IMS, Universität Stuttgart.

Carroll, G. und M. Rooth (1998). Valence induction with a head-lexicalized PCFG. In *Proceedings of EMNLP-3*, Granada.

Charniak, E. (1993). *Statistical Language Learning*. Cambridge, MA: M.I.T. Press.

Charniak, E. (1995). Parsing with context-free grammars and word statistics. Technical Report CS-95-28, Department of Computer Science, Brown University.

Charniak, E. (1996). Tree-bank grammars. Technical Report CS-96-02, Brown University.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the 14th AAAI*, Menlo Park, CA.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics (ANLP-NAACL 2000)*, Seattle, WA.

Chelba, C. (1997). A structured language model. In *Proceedings of ACL-EACL*, Madrid, Spain.

Chelba, C., D. Engle, F. Jelinek, V. Jiminez, S. Khudanpur, L. Mangu, H. Printz, E. Ristad, R. Rosenfeld, A. Stolcke, und D. Wu (1997). Structure and performance of a dependency language model. In *Proceedings of Eurospeech*, Rhodes, Greece.

Chelba, C. und F. Jelinek (1998). Exploiting syntactic structure for language modeling. In *Proceedings of COLING-ACL*, Montreal, Canada.

Chelba, C. und F. Jelinek (2000). Structured language modeling. *Computer Speech and Language*.

Chi, Z. (1998). Estimation of probabilistic context-free grammars. *Computational Linguistics 24*(2).

Chi, Z. (1999). Statistical properties of probabilistic context-free grammars. *Computational Linguistics 25*(1).

Ciaramita, M. und M. Johnson (2000). Explaining away ambiguity: Learning verb selectional preference with bayesian networks. In *Proceedings of COLING-2000*, Saarbrücken.

Clarkson, P. und R. Rosenfeld (1997). Statistical language modeling using the CMU-Cambridge toolkit. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*.

Collins, M. (1996). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL'96)*, Santa Cruz, CA.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL'97)*, Madrid.

Collins, M. und S. Sekine (2001). Scoring code for the Wall Street Journal Penn treebank. http://www.research.att.com/~mcollins/.

Cormen, T. H., C. E. Leiserson, und R. L. Rivest (1994). *Introduction to Algorithms*. Cambridge, MA: The MIT Press.

Cover, T. M. und J. A. Thomas (1991). *Elements of Information Theory*. New York: Wiley.

Dagan, I. und A. Itai (1994). Word sense disambiguation using a second language monolingual corpus. *Computational Linguistics 20*, 563–596.

Dagan, I., L. Lee, und F. Pereira (1998). Similarity-based models of word cooccurrence probabilities. To appear in *Machine Learning*.

Darroch, J. und D. Ratcliff (1972). Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics 43*(5), 1470–1480.

DeGroot, M. H. (1989). *Probability and statistics* (2 ed.). Addison-Wesley.

Della Pietra, S., V. Della Pietra, V. Gillet, J. Lafferty, J. Printz, und H. Ures (1994). Inference and estimation of a long-range trigram model. Technical report, School of Computer Science, CMU.

Della Pietra, S., V. Della Pietra, und J. Lafferty (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*(4), 380–393.

Dempster, A. P., N. M. Laird, und D. B. Rubin (1977). Maximum likelihood from incomplete data via the *EM* algorithm. *J. Royal Statist. Soc. 39*(B), 1–38.

Dotzauer, E. (1987). *Grundlagen der digitalen Simulation*. München: Carl Hanser Verlag.

Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics 19*(1).

Elworthy, D. (1994). Does Baum-Welch re-estimation help taggers? In *Proceedings of the 4th ACL Conference on Applied Natural Language Processing*, Stuttgart.

Ersan, M. und E. Charniak (1995). A statistical syntactic disambiguation program and what it learns. Technical Report CS-95-29, Department of Computer Science, Brown University.

Erwe, F. (1962). *Differential- und Integral-Rechnung, Band 1-2.* BI Hochschul-Taschenbücher.

Evert, S., U. Heid, und W. Lezius (2000). Methoden zum Vergleich von Signifikanzmasen zur Kollokationsidentifikation. In *Proceedings of KONVENS'2000*, Ilmenau.

Fischer, G. (1980). *Lineare Algebra.* Braunschweig: Vieweg.

Forster, O. (1981). *Analysis. Band 1-2.* Braunschweig: Vieweg.

Frank, A., T. H. King, J. Kuhn, und J. Maxwell (1998). Optimality theory style constraint ranking in large-scale LFG grammars. In *Proceedings of the Third LFG Conference.*

Frey, B. J. (1998). *Graphical Models for Machine Learning and Digital Communication.* Cambridge, MA: The MIT Press.

Gao, J. und K.-F. Lee (2000). Distribution-based pruning of backoff language models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00)*, Hong Kong, China.

Goodman, J. (1998). *Parsing Inside-Out.* Ph. D. thesis, Computer Science Group, Harvard University, Cambridge, MA.

Heid, U. (1994). On ways words work together - topics in lexical combinatorics. In *Proceedings of the* EURALEX *International Conference*, Amsterdam.

Heid, U. (1998). Towards a corpus-based dictionary of german noun-verb collocations. In *Proceedings of the* EURALEX *International Conference*, Amsterdam.

Helbig, G. (1984). Probleme der Beschreibung von Funktionsverbgefügen im Deutschen. In G. Helbig (Ed.), *Studien zur deutschen Syntax*, Volume 2. Leipzig.

Helbig, G. und J. Buscha (1996). *Deutsche Grammatik. Ein Handbuch für den Ausländerunterricht.* Leipzig: Langenscheidt.

Hofmann, T. und J. Puzicha (1998). Unsupervised learning from dyadic data. Technical Report TR-98-042, ICSI, Berkeley, CA.

Hopcroft, J. E. und J. D. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation.* Reading, MA: Addison-Wesley.

Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical Review 106*, 620–630.

Jelinek, F. (1997). *Statistical Methods for Speech Recognition.* Cambridge, Massachusetts: MIT Press.

Johnson, M. (1988). *Attribute-value Logic and the Theory of Grammar.* CSLI lecture notes 16. Stanford, CA: CSLI.

Johnson, M. (2001a). *An Implementation of The Inside-Outside Algorithm.* http://www.cog.brown.edu/~mj.

Johnson, M. (2001b). Trading recall for precision with confidence sets. Submitted.

Johnson, M., S. Geman, S. Canon, Z. Chi, und S. Riezler (1999). Estimators for stochastic "unification-based" grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, College Park, MD.

Johnson, M. und S. Riezler (2000). Exploiting auxiliary distributions in stochastic unification-based grammars. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics (ANLP-NAACL 2000)*, Seattle, WA.

Jurawsky, D. und J. H. Martin (2000). *Speech and Language Processing.* Prentice Hall.

Kamp, H. (1999). Können Sie mit Ihren Modellen auch Kollokationen erkennen ? personal communication.

Karttunen, L., T. Yampol, und G. Grefenstette (1994). *INFL Morphological Analyzer/Generator 3.2.9 (3.6.4).* Xerox Corporation.

Kasami, T. (1965). An efficient recognition and syntax algorithm for context-free languages. Technical report, Air Force Cambridge Research Lab.

Khudanpur, S. und J. Wu (2000). Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling. *Computer Speech and Language.*

Kilgarriff, A. (1996). Which words are particulary characteristic of a text? a survey of statistical approaches. In *Proceedings of AISB Workshop on Language Engineering for Document Analysis and Recognition.*

Kilgarriff, A. und J. Rosenzweig (2000). English SENSEVAL: Report and results. In *Proceedings of LREC 2000.*

Kim, K. und K.-S. Choi (2000). Dimension-reduced estimation of word co-occurrence probability. In *Proc. of the ACL'00.*

Kneser, R. und H. Ney (1993). Improved clustering techniques for class-based statistical language modeling. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech).*

Krengel (1987). *Einführung in die Wahrscheinlichkeitstheorie.* Vieweg.

Krenn, B. und C. Samuelsson (1997). The linguist's guide to statistics. http://www.coli.uni-sb.de/{~krenn,~christer}.

Kuhn, J. (2000). Processing optimality-theoretic syntax by interleaved chart parsing and generation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00)*, Hong Kong.

Lafferty, J. D. (1993). A derivation of the inside-outside algorithm from the EM algorithm. IBM Technical Report.

Lari, K. und S. J. Young (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language 4*, 35–56.

Lari, K. und S. J. Young (1991). Applications of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language 5*, 237–257.

LDC (1993). Penn wall street journal treebank [cd-rom]. Linguistic Data Consortium, University of Pennsylvania. e-mail: ldc@unagi.cis.upenn.edu.

Levin, B. (1993). *English Verb Classes and Alternations. A Preliminary Investigation.* Chicago/London: The University of Chicago Press.

Lezius, W. (1999). Automatische Extrahierung idiomatischer Bigramme aus Textkorpora. In *Tagungsband des Linguistischen Kolloquiums*, Germersheim.

Magerman, D. M. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL'95)*, Cambridge, MA.

Manning, C. und H. Schütze (1999). *Foundations of Statistical Natural Language Processing.* MIT Press.

Marcus, M. P., B. Santorini, und M. A. Marcinkiewicz (1993). Building a large annotated corpus of english: The Penn treebank. *Computational Linguistics 19*(2), 313–330.

Maxwell, J. und R. Kaplan (1996). Unification-based parsers that automatically take advantage of context freeness. Unpublished manuscript, Xerox Palo Alto Research Center.

McCallum, A. K. (1996). Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. `http://www.cs.cmu.edu/ mccallum/bow`.

McLachlan, G. J. und T. Krishnan (1997). *The EM Algorithm and Extensions.* New York: Wiley.

Mitchell, T. M. (1997). *Machine Learning.* Singapore: McGraw Hill.

Müller, K. (2000). Probabilistische kontext-freie Grammatiken für Silbifizierung und Graphem-Phonem-Konvertierung. In *Research Papers from the Phonetics Lab*, AIMS Report 6(4). Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Müller, K. (2001a). Automatic detection of syllable boundaries combining the advantages of treebank and bracketed corpora training. In *Proceedings of ACL'01*, Toulouse.

Müller, K. (2001b). Probabilistic context-free grammars for syllabification and grapheme-to-phoneme conversion. In *Proceedings of EMNLP'01*, Pittsburgh.

Müller, K. (to appear 2001). *Probabilistic Syllable Modeling Using Probabilistic Context-Free Grammars and EM-Based Clustering*. Ph. D. thesis, IMS, University of Stuttgart, Stuttgart.

Müller, K., B. Möbius, und D. Prescher (2000a). Inducing probabilistic syllable classes using multivariate clustering. In *Proc. of ACL-2000*.

Müller, K., B. Möbius, und D. Prescher (2000b). Inducing probabilistic syllable classes using multivariate clustering - GOLD. In *AIMS Report 6(2)*, IMS, Univ. Stuttgart.

Ngai, G. und D. Yarowsky (2000). Rule writing or annotation: Cost-efficient resource usage for base noun phrase chunking. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00)*.

Ostrowski, A. M. (1966). *Solution of Equations and Systems of Equations* (2 ed.). New York: Academic.

Pereira, F. und Y. Schabes (1992). Inside-outside reestimation from partially bracketed corpora. In *Proceedings of ACL'92*, Newark, Delaware.

Pereira, F., N. Tishby, und L. Lee (1993). Distributional clustering of English words. In *Proc. of ACL'93*.

Prescher, D. (1999). *Manual pages for* `hsc` *and* `hscc`. IMS, Universität Stuttgart.

Prescher, D. (2000). EM&NLP. Manuscript. IMS, Universität Stuttgart.

Prescher, D. (2001b). Inside-outside estimation meets dynamic EM. In *Proceedings of IWPT-2001*, Beijing.

Prescher, D. (2001c). Inside-outside estimation meets dynamic EM — GOLD. Technical report, DFKI GmbH, Language Technology Lab.

Prescher, D. (2001d). Novel properties and well-tried performance of EM-based multivariate clustering. In *Proc. of RANLP-2001*.

Prescher, D. (to appear 2001a). EMMC Toolkit. Software package for EM-based multivariate clustering. IMS, Universität Stuttgart. `http://www.ims.uni-stuttgart/~prescher`.

Prescher, D. und U. Heid (2000). Probabilistisches Clustering zur Identifikation von Verb-Nomen-Kollokationen. Presented at DGfS 2000, Marburg.

Prescher, D., S. Riezler, und M. Rooth (2000). Using a probabilistic class-based lexicon for lexical ambiguity resolution. In *Proc. of COLING-2000*.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, und B. P. Flannery (1992). *Numerical Recipes in C: The Art of Scientific Computing*. New York: Cambridge University Press.

Ratnaparkhi, A. (1997). A linear observed time statistical parser based on maximum entropy models. In *Proceedings of EMNLP-2*.

Resnik, P. (1993). *Selection and information: A class-based approach to lexical relationships*. Ph. D. thesis, University of Pennsylvania, CIS Department.

Resnik, P. (1997). Selectional preference and sense disambiguation. In *Proceedings of the ANLP'97 Workshop: Tagging Text with Lexical Semantics: Why, What, and How?*, Washington, D.C.

Ribas, F. (1994). An experiment on learning appropriate selectional restrictions from a parsed corpus. In *Proceedings of COLING-94*, Kyoto, Japan.

Riezler, S. (1999). *Probabilistic Constraint Logic Programming*. Ph. D. thesis, Seminar für Sprachwissenschaft, Universität Tübingen. AIMS Report, 5(1), IMS, Universität Stuttgart.

Riezler, S., D. Prescher, J. Kuhn, und M. Johnson (2000). Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and EM training. In *Proc. of ACL-2000*.

Rooth, M. (1995). Two-dimensional clusters in grammatical relations. In *Symposium on Representation and Acquisition of Lexical Knowledge: Polysemy, Ambiguity, and Generativity*.

Rooth, M. (1998). Two-dimensional clusters in grammatical relations. In *Inducing Lexicons with the EM Algorithm*, AIMS Report 4(3). Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Rooth, M., S. Riezler, D. Prescher, G. Carroll, und F. Beil (1998). EM-based clustering for NLP applications. In *Inducing Lexicons with the EM Algorithm*, AIMS Report 4(3). Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Rooth, M., S. Riezler, D. Prescher, G. Carroll, und F. Beil (1999). Inducing a semantically annotated lexicon via EM-based clustering. In *Proc. of ACL'99*.

Rosenfeld, R. (2000). Two decades of statistical language modelling: Where do we go from here? In *Proceedings of the IEEE*, Volume 88.

Saul, L. K. und F. Pereira (1997). Aggregate and mixed-order Markov models for statistical language processing. In *Proceedings of EMNLP-2*.

Schiller, A. und C. Stöckert (1995). *DMOR*. IMS, Universität Stuttgart.

Schmid, H. (1999a). *LoPar. Design and Implementation*. Insitut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Schmid, H. (1999b). *YAP: Parsing and Disambiguation with Feature-Based Grammars*. Ph. D. thesis, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Schuhmacher, H. (1986). *Verben in Feldern. Valenzwörterbuch zur Syntax und Semantik deutscher Verben*. Berlin: de Gruyter.

Schulte im Walde, S. und H. Schmid (2000). Robust german noun chunking with a probabilistic context-free grammar. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*.

Schütze, H. (1992). Dimensions of meaning. In *Proceedings of Supercomputing '92*.

Shannon, C. E. (1949). A mathematical theory of communication. *The Mathematical Theory of Communication*.

Skut, W. und T. Brants (1998). A maximum-entropy partial parser for unrestricted text. In *Proceedings of the Sixth Workshop on Very Large Corpora*, Montréal.

Stolcke, A. (1999). SRILM – the SRI language modeling toolkit. `http://www.speech.sri.com/projects/srilm`.

Van der Warden, B. (1936). *Algebra*. Berlin: Springer.

Wu, C. F. J. (1983). On the convergence properties of the EM algorithm. *The Annals of Statistics 11*(1), 95–103.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL'95*, Cambridge.

Yeh, A. (2000). More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th international conference on computational linguistics (COLING 2000)*, Saarbrücken.

Zangwill, W. I. (1969). *Nonlinear Programming: A Unified Approach*. Englewood Cliffs, New Jersey: Prentice Hall.